



Dissertation

Master in Computer Engineering – Mobile Computing

***DASHBOARD DESIGN TO ASSESS THE IMPACT
OF DISTINCT DATA VISUALIZATION
TECHNIQUES IN THE DYNAMIC ANALYSIS OF
SURVEY'S RESULTS***

Renato Mauricio Toasa Guachi

Leiria, June 2018

This page was intentionally left blank



***DASHBOARD DESIGN TO ASSESS THE IMPACT
OF DISTINCT DATA VISUALIZATION
TECHNIQUES IN THE DYNAMIC ANALYSIS OF
SURVEY'S RESULTS***

Renato Mauricio Toasa Guachi

Dissertation developed under the supervision of Doctor Marisa da Silva Maximiano, Professor at the School of Technology and Management of the Polytechnic Institute of Leiria, co-directed by Doctor Catarina Isabel Ferreira Viveiros Tavares dos Reis, Professor at the School of Technology and Management of the Polytechnic Institute of Leiria and co-directed by Ingeniero David Omar Guevara Aulestia, Professor at the Universidad Técnica de Ambato.

Leiria, June 2018

This page was intentionally left blank

Dedication

A mis padres José Luis y Norma que han sabido formarme con valores y buenos sentimientos, los cuales me han ayudado a salir adelante en los momentos difíciles que me ha tocado vivir.

A mis abuelitos que siempre han estado junto a mí brindándome su apoyo y consejos con lo que he logrado afrontar los retos que se me han presentado a lo largo de mi vida. Y a los que no están presentes pero sé que siempre me cuidan desde el cielo.

A mis tíos, a mis padrinos Hilda y Carmelo que gracias a su ayuda desinteresada y sabios consejos logre culminar con mi meta trazada.

Y esta dedicado a todas las personas que de alguna manera fueron participes de mi desarrollo académico profesional y personal.

This page was intentionally left blank

Acknowledgements

Agradezco infinitamente a mis tutores Dra. Marisa Maximiano, Dra. Catarina Reis, Ing David Guevara que con su ayuda, conocimientos, exigencias, consejos y experiencia me permitieron seguir adelante para de esta forma lograr culminar con este difícil escalón hacia la vida profesional.

De igual manera mi sincero agradecimiento a La Secretaria de Educación Superior, Ciencia, Tecnología e Innovación (Senescyt) de Ecuador por financiar mis estudios de Maestría, y a la Universidad Técnica de Ambato por el apoyo brindado en el proceso de estudios de Maestría.

This page was intentionally left blank

Resumo

É necessário realizar um estudo sobre as técnicas de visualização de dados disponíveis para exibir de forma compreensiva e precisa, informações em tempo real. Personalizar plataformas já existentes e projetar novos formatos específicos de visualização estão entre as tarefas importantes para obter uma visualização precisa das informações.

Neste trabalho realizámos uma revisão bibliográfica sobre visualização de dados, técnicas e plataformas de Dashboards existentes. Implementámos um dashboard genérico e dinâmico com base em informações recolhidas em tempo real. O objetivo foi o de avaliar o impacto das Técnicas de Visualização de Dados disponíveis no dashboard desenvolvido.

Portanto, os utilizadores do Dashboard poderão interagir com a informação, acedendo numa fase inicial a um conjunto de gráficos, tabelas e relatórios pré-definidos, produzidos pelo próprio Dashboard. Numa fase posterior será possível aos utilizadores ter um controlo maior sobre a apresentação dos dados e personalizar as vistas apresentadas, gerando gráficos, tabelas e relatórios dinamicamente.

Esta implementação permite testar um conjunto existente de técnicas de visualização de dados e as novas formas geradas dinamicamente, mostrando que os Dashboards se podem tornar um meio exclusivo e poderoso de fornecer informação.

Palavras-chave: Técnicas de visualização de dados; painel de controle; informação em tempo real; *dashboard* personalizado;

This page was intentionally left blank

Abstract

A study of available data visualization techniques is required to comprehensively and accurately display real-time information. Customizing existing platforms and designing new specific display formats are among the important tasks for getting an accurate view of information. In this work we have carried out a bibliographic review on visualization of data, techniques and platforms of existing Dashboards. We implemented a generic and dynamic dashboard based on information collected in real time. The objective was to assess the impact of the Data Visualization Techniques available on the developed dashboard.

Therefore, Dashboard users will be able to interact with the information, accessing at an early stage a set of pre-defined charts, tables, and reports produced by the Dashboard itself. At a later stage it will be possible for users to have greater control over the presentation of the data and to customize the views presented, generating graphs, tables and reports dynamically.

This implementation allows you to test an existing set of data visualization techniques and dynamically generated new forms, showing that Dashboards can become a unique and powerful means of providing information.

Keywords: Data visualization techniques; Dashboard; real-time information; platform; tailored

This page was intentionally left blank

Resumen

Es necesario realizar un estudio sobre las técnicas de visualización de datos disponibles para mostrar de forma comprensiva y precisa, informaciones en tiempo real. Personalizar plataformas ya existentes y diseñar nuevos formatos específicos de visualización están entre las tareas importantes para obtener una visualización precisa de la información. En este trabajo realizamos una revisión bibliográfica sobre visualización de datos, técnicas y plataformas de Dashboards existentes. Hemos implementado un Dashboard genérico y dinámico basado en información recopilada en tiempo real. El objetivo fue evaluar el impacto de las técnicas de visualización de datos disponibles en el tablero de instrumentos desarrollado.

Por lo tanto, los usuarios del Dashboard pueden interactuar con la información, accediendo en una fase inicial a un conjunto de gráficos, tablas e informes predefinidos, producidos por el propio Dashboard. En una fase posterior, los usuarios pueden tener un mayor control sobre la presentación de los datos y personalizar las vistas presentadas, generando gráficos, tablas e informes dinámicamente.

Esta implementación permite probar un conjunto existente de técnicas de visualización de datos y las nuevas formas generadas dinámicamente, mostrando que los Dashboards pueden convertirse en un medio exclusivo y poderoso de proporcionar información.

Palabras Clave: Técnicas de Visualización de Datos; Dashboard; Información en tiempo real; Plataforma personalizada

List of figures

Figure 1 - Data Information, knowledge and decisions. 1

Figure 2 - Systematic Literature Review steps [5]. 5

Figure 3 - Results – Systematic Review..... 7

Figure 4 - A word cloud shows the words or phrases associated with a topic. 13

Figure 5 - Network diagrams explore relationships within a data set, including connections across geographic areas. 14

Figure 6 - In this correlation matrix, darker boxes indicate a stronger correlation; lighter boxes indicate a weaker correlation. 15

Figure 7 - Autocharting in SAS Visual Analytics produces a bar chart to show the distribution of a single measure. 15

Figure 8 - A Sankey diagram displays a series of linked nodes, where the width of each node indicates the frequency of the link or value of the measure. 16

Figure 9 - Grid design translated to different screens. 17

Figure 10 - Pedagogical questionnaires. 19

Figure 11 - Questionnaire on patient satisfaction..... 20

Figure 12 – Generic Data Model..... 22

Figure 13 – General Architecture with the technology used..... 23

Figure 14 – Menu and Statistics view sections. 29

Figure 15 - Temporal Analysis..... 29

Figure 16 - Home Section. 30

Figure 17 - Dynamic Chart section. 31

Figure 18 - Surveys section..... 31

Figure 19 - Text Survey. 32

Figure 20 – Answers of Numeric Survey..... 32

Figure 21 - Statistics of the Answers of Numeric Survey in a chart..... 32

Figure 22 - Statistics of the Answers of Numeric Survey in a table. 33

Figure 23 - Users Section. 33

Figure 24 - Answers Section. 33

Figure 25 - Question Types Section..... 34

Figure 26 - Questions Section. 34

Figure 27 - Migration structure, Table users..... 36

Figure 28 - Seeder Structure, table user.	37
Figure 29 - Model factory of the tables.	37
Figure 30 - Seeders Call	38
Figure 31 - Data created in the table.....	38
Figure 32 - Model Structure.	39
Figure 33 - Controllers Structure.....	40
Figure 34 - Even Service Provide.....	41
Figure 35 - Event and Listener.	42
Figure 36 – Views in the project structure.	43
Figure 37 - Main Dashboard view.....	43
Figure 38 - Responsive Design.....	44
Figure 39 - Date picker for data range.....	44
Figure 40 - Carbon Function.	44
Figure 41 - Searh, Sort and Pagination.....	45
Figure 42 - Line chart of Questions by User.	45
Figure 43 - Function to graph the Line chart.....	46
Figure 44 - Pie chart of Question Types.....	46
Figure 45 - Function to graph the Pie chart.	47
Figure 46 - Radar chart to show the Surveys Duration.	48
Figure 47 - Function to graph the Radar chart.	48
Figure 48 - Bar chart to show the Surveys Count.....	49
Figure 49 - Function to graph the Bar chart.	49
Figure 50 - Doughnut Chart to show the anonymous and non-anonymous surveys.....	50
Figure 51 - Function to graph the Doughnut Chart.	50
Figure 52 - Socket.js content for communication.	51
Figure 53 - Function to connect.....	51
Figure 54 - Literature Review Results.....	71
Figure 55 - Prototype - User View.	75
Figure 56 - Prototype - Answer View.	75
Figure 57 - Prototype - Questions View.....	76
Figure 58 - Prototype - Surveys View.....	76
Figure 59 - Xampp Control Panel.	77
Figure 60 - Laravel dependences.....	78
Figure 61 - Database.php file.	79

Figure 62 - .env file..... 80

This page was intentionally left blank

List of tables

Table 1 - Dashboard Platforms Comparison..... 12

Table 2 - Test Results..... 57

Table 3 - Reference Papers..... 70

This page was intentionally left blank

List of acronyms

API	Application Programming Interface.
APP	Application
BI	Business Intelligence
CLI	Command Line Interface
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
CRM	Customer Relationship Management
DB	Database
HTML	Hyper Text Markup Language
JS	Javascript
JSON	JavaScript Object Notation
IBM	International Business Machines Corporation
I/O	Input/ Output
KPI	Key Performance Indicator
MVC	Model View Controller
OS	Operating System
PHP	Hypertext Preprocessor
RWD	Responsive Web Design
SAS	Analytics Software & Solutions
SLR	Systematic Literature Review
URL/URI	Uniform Resource Locator / Identifier
UI	User Interface
WWW	World Wide Web
XML	Extended Markup Language

This page was intentionally left blank

Table of Contents

DEDICATION	III
ACKNOWLEDGEMENTS	V
RESUMO	VII
ABSTRACT	IX
RESUMEN	XI
LIST OF FIGURES	XII
LIST OF TABLES	XV
LIST OF ACRONYMS	XVII
TABLE OF CONTENTS	XIX
1. INTRODUCTION	1
1.1. Problem Description and Motivation	1
1.2. Research Contribution.....	2
1.3. Goals.....	3
1.4. Organization of the Document	3
2. LITERATURE REVIEW	5
2.1. Methodology – Systematic Literature Review	5
2.2. Dashboards	8

2.2.1.	Type of Dashboards	9
2.2.2.	Dashboard Platforms	10
2.2.3.	Data Visualization Techniques	13
3.	PROJECT OUTLINE	19
3.1.	Case Studies.....	19
3.2.	Data Model	21
3.3.	Technologies and tools	22
	Laravel.....	23
	MySQL.....	24
	PHP	24
	Artisan Console.....	24
	Composer	24
	Javascript Libraries	24
	Chart JS	25
	Highcharts JS	25
	Bootstrap	25
	Jquery.min.js	25
	Carbon API	25
	Redis.....	26
	Node.js	26
	Socket. IO.....	26
	Font Awesome	26
	Responsive Web Design	26
	Operating System	26
	XAMPP	27
	Sublime Text	27
	Git.....	27
	Git-Hub	27
4.	DASHBOARD IMPLEMENTATION	29
4.1.	Dashboard Sections and Features	29

4.2.	Implementation Details	34
4.2.1.	Migrations	35
4.2.2.	Seeders	36
4.2.3.	Model factory	37
4.2.4.	Models	38
4.2.5.	Controllers	39
4.2.6.	Events	40
4.2.7.	Views	42
4.2.8.	Date picker to Temporal Analysis	44
4.2.9.	Date format	44
4.2.10.	Search, Sort and Pagination	45
4.2.11.	Data Visualization used	45
4.2.12.	Real Time Communication	50
4.3.	Security Considerations	52
5.	PRODUCTION ENVIRONMENT, USER TESTING AND RESULTS	55
5.1.	Project Web Hosting – Production Environment	55
5.2.	Test and Results	55
6.	CONCLUSIONS AND FUTURE WORK	59
7.	REFERENCES	60
	APPENDICES	68
	Appendix A	68
	Appendix B	72
	Appendix C	73
	Appendix D	74
	Appendix E	75

Appendix F.....	77
Appendix G.....	81
Appendix H.....	82
GLOSSARY	84

This page was intentionally left blank

1. Introduction

In this chapter the search of work, articles, journals are carried out where several generic concepts referring to the visualization of data are mentioned. Based on this information, the description of the problem and the motivation that allows the development of this research work is made.

1.1. Problem Description and Motivation

In Visualization, data, information and knowledge are three terms used extensively, often in an interrelated context. In many cases, they are used to indicate different levels of abstraction, understanding or truthfulness [1]. For example, Visualization is concerned with exploring data and information, and the primary objective in data visualization is to gain insight into an information space and the visualization of the information is for the mining of data and the discovery of knowledge, allowing the user to make correct decisions.

The information is data that has been given meaning by way of relational connection, while the knowledge is the appropriate collection of information, such that it's intent is to be useful. Knowledge is a deterministic process which allows decisions to be made [2]. Figure 1 presents the relationship between these concepts.

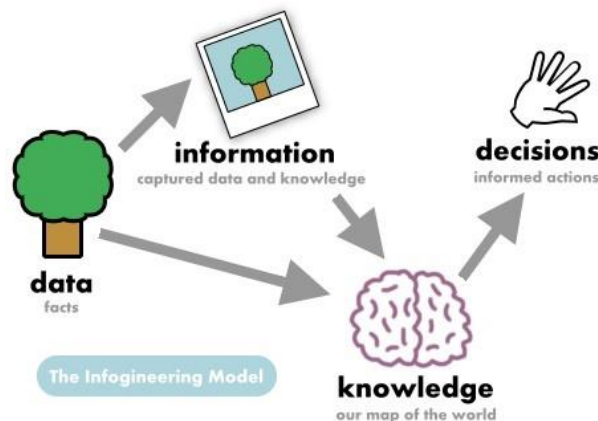


Figure 1 - Data Information, knowledge and decisions.

Information design is the art and science of integrating writing and design, so that people can use content in ways that suit their personal goals. Information design involves making communication artifacts by shaping verbal language and visual language [3]. Artifacts are physical objects, such as clothing, homes, and cars, that

indicate to others a person's personal and social beliefs and habits. Messages are thus conveyed in a nonverbal manner [4].

Infovis (Information visualization)is traditionally viewed as a tool for data exploration and hypothesis formation. Because of its roots in scientific reasoning, visualization has traditionally been viewed as an analytical tool for sensemaking. In recent years, however, both the mainstreaming of computer graphics and the democratization of data sources on the Internet have had important repercussions in the field of information visualization. With the ability to create visual representations of data on home computers, artists and designers have taken matters into their own hands and expanded the conceptual horizon of Information visualization as artistic practice [5].

The combination of infovis and mathematical deduction to extract patterns in massive, dynamically changing information spaces has, of late, become one of the main themes of the visualization academic community. As with research that supports scientific visualization, the emphasis has been on visualization as a tool for dispassionate analysis [5].

This relentless focus on visualization as a neutral tool may appear to be a forced move. At first, bias in a visualization might seem like a technical problem, much like chromatic aberration in a telescope. Yet a recent, separate stream of thought in visualization calls this assumption into question [5].

Due to the massive amount of information that exists and is created every day, the number of existing artifacts and the needs of all users and consumers of knowledge that exist, it is difficult to find ways to present the information in an accessible way or make sense. This is when the need arises to implement a tool that fulfills these functions. This type of tool, called Dashboard, is mainly aimed at technology managers of organizations to plan, design, implement and direct applications in real time, and thus monitor the information.

1.2. Research Contribution

There are several dashboards available in the market that are used by large companies for the visualization and analysis of their information, this is detailed in the following chapter. The majority of available dashboards are not free, so the user must pay for their services, also these has a general approach, when applied to surveys should be

configured by some expert. For these reasons it is very important to develop a free Dashboard where its focus is the analysis of surveys, since the surveys generate information of great value for the making of decisions in very different contexts, becoming relevant for all the companies of the world.

1.3. Goals

This work has been designed and implemented in order to be able to accomplish the following goals:

- i. The objective is to produce a dynamic Board for the analysis of surveys. Therefore, Dashboard users will be able to interact with the information, based on an initial set of data, tables, graphs produced by the Board itself. This will allow testing an existing set of visualization data tools and techniques and discover a new set of visualization data tools and techniques;
- ii. Conduct a core state of the art about Data Visualization techniques;
- iii. Identify a set of distinct Data Visualization techniques to explore in the dynamic and real-time context;
- iv. Design and implement a Dashboard solution for the context;

1.4. Organization of the Document

The present research is organized as follows: Chapter 2 succinctly describes literature review and state of the art of dashboards, Chapter 3 shows project outline, Chapter 4 presents Dashboard implementation. Next, Chapter 5 describes production environment, user testing and results and finally conclusions and future work are detailed in Chapter 6.

This page was intentionally left blank

2. Literature Review

This chapter describes the methodology used in the search for related work. The 5 steps of Systematic literature review are developed, oriented to the objectives of this research. In addition, the state of the art of this research is made.

2.1. Methodology – Systematic Literature Review

Before starting with the proposed research, it was necessary to make a Systematic Review of “Dashboard Design”, “Data Visualization Techniques” in the “Dynamic Analysis of Real-Time Information”.

In this research we conducted a Systematic Literature Review (SLR) [6] as a means to structure and organize the review. It is formed by 5 steps (see Figure 2).

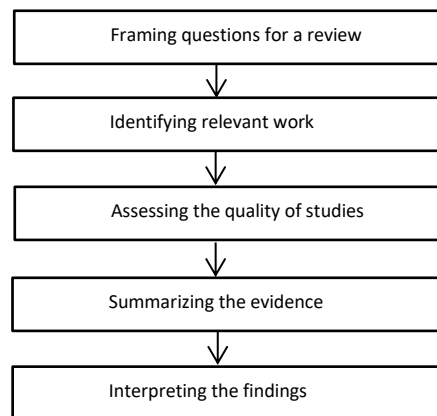


Figure 2 - Systematic Literature Review steps [5].

Step 1. Framing questions for a review

The problems to be addressed by the review should be specified in the form of clear, unambiguous and structured questions before beginning the review work. Once the review questions have been set, modifications to the protocol should be allowed only if alternative ways of defining the populations, interventions, outcomes or study designs become apparent.

Step 2. Identifying relevant work

The search for studies should be extensive. Multiple resources should be searched without language restrictions. The study selection criteria should flow directly from the review questions and be specified a priori. Reasons for inclusions should be recorded.

Step 3. Assessing the quality of studies

Study quality assessment is relevant to every step of a review. Question formulation (Step 1) and study selection criteria (Step 2) should describe the minimum acceptable level of design. Selected studies should be subjected to a more refined quality assessment by use of general critical appraisal guides and design-based quality checklists (Step 3). These detailed quality assessments will be used for exploring heterogeneity and informing decisions regarding suitability of meta-analysis (Step 4). In addition, they help in assessing the strength of inferences and making recommendations for future research (Step 5).

Step 4. Summarizing the evidence

Data synthesis consists of tabulation of study characteristics, quality and effects as well as use of statistical methods for exploring differences between studies and combining their effects. Exploration of heterogeneity and its sources should be planned in advance (Step 3). If an overall meta-analysis cannot be done, subgroup meta-analysis may be feasible.

Step 5. Interpreting the findings

The issues highlighted in each of the four steps above should be met. The risk of publication bias and related biases should be explored. Exploration for heterogeneity should help determine whether the overall summary can be trusted, and, if not, the effects observed in high-quality studies should be used for generating inferences. Any recommendations should be graded by reference to the strengths and weaknesses of the evidence

The summary for result of these steps is show in the Figure 3.

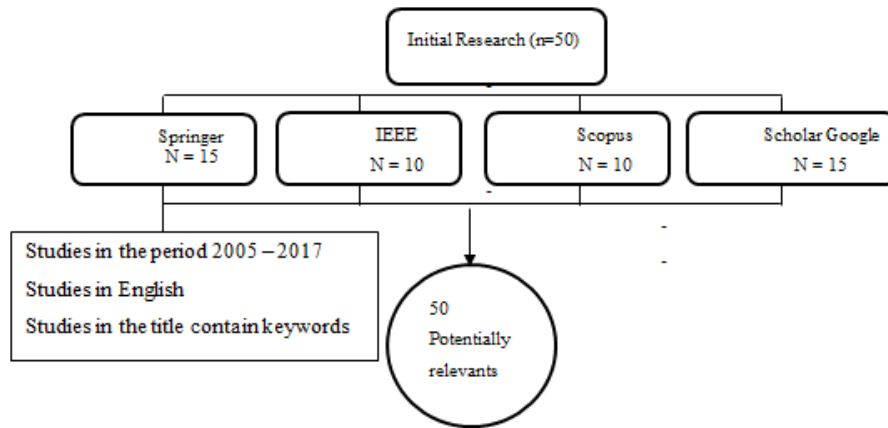


Figure 3 - Results – Systematic Review.

Systematic reviews are types of literature reviews that collect and critically analyze multiple research studies or papers, using methods that are selected before one or more research questions are formulated, and then finding and analyzing studies that relate to and answer those questions in a structured methodology. In this case information was obtained as a starting point for carrying out this research.

As a result, we obtained 50 relevant documents as papers, books, chapters, on the subject of this research indexed in high impact academic databases and that they meet all the steps of the systematic review, which will serve as a starting point for the development of this research.

This revision is important since it allows to know in depth the concepts and techniques related to the visualization of information and the development of dashboards, allowing us to have a clear objective in the development of this research.

The table summarizing all the findings which contains detailed information on the investigations explored and analyzed is detailed in Appendix A.

2.2. Dashboards

The information can be presented through the use of suitable Dashboards, adequately designed for the context at hand. A Dashboard is the graphical and visual representation of information.

Dashboards are nowadays widely used for monitoring and analysis of business processes. Numerous companies such as IBM , SAP, Tableau Software or TIBCO Spotfire, to name a few well-known vendors, offer complete Business Intelligence (BI) or information visualization solutions [7].

Dashboards can provide a unique and powerful means to present information, but they rarely live up to their potential. Most Dashboard fail to communicate efficiently and effectively, not because of inadequate technology, but because of poorly designed implementations. No matter how great the technology, a Dashboard's success as a medium of communication is a product of design, a result of a display that speaks clearly and immediately. Dashboard can tap into the tremendous power of visual perception to communicate, but only if those who implement them understand visual perception and apply that understanding through design principles and practices that are aligned with the way people see and think [8].

Dashboards and visualization are cognitive tools that improve your “span of control” over a lot of business data. These tools help people visually identify trends, patterns and anomalies, reason about what they see and help guide them toward effective decisions. As such, these tools need to leverage people’s visual capabilities. With the prevalence of scorecards, Dashboards and other visualization tools now widely available for business users to review their data, the issue of visual information design is more important than ever [9].

Before delving too deeply into what Dashboard software is capable of doing, it is important to figure out what you need it to do for your work. Next, it is presented a general overview of the types of Dashboard software available [10].

2.2.1. Type of Dashboards

Nowadays, different types of Dashboard software are available, mainly:

- **BI Dashboards** - This software usually offers a suite of Dashboards that focus on specific areas of the business such as analytics, strategy, operations, overall performance, and Key Performance Indicator (KPI). Also find tools for monitoring parts of the business and for gathering informational data that tells a story [10];
- **KPI Dashboard Software** - Often found as an element in BI Dashboards, this is a Dashboard that focuses solely on delivering and analyzing key performance metrics. It can also be used to measure performance within a team [10];
- **Website Dashboards** - There are Dashboards specifically designed to monitor metrics and display them at a glance. They typically allow the user to add widgets for services like Google Analytics, GitHub, Pingdom, and Chartbeat to Dashboard software that monitors website performance. It also allows creating custom data sources and incorporating social media monitoring [10];
- **Free Dashboards** - There are a variety of open source Dashboard software options that you can use to build your own Dashboards. Many of the open source options are web-based, but there are a few that are server-based. Additionally, these Dashboards are ideal for integrating custom data streams. However, there typically isn't a ton of support available [10];
- **Free Dashboards** - There are plenty of free options available. However, the free solutions are not as robust as other options available and have limited features. Free Dashboard software is a good place to start, to realize the power of real-time data analytics applied to a business [11];
- **Marketing Dashboards** - This is another type of Dashboard that might be found in a BI Dashboard. These tend to focus on how well marketing efforts are increasing sales or building awareness. Marketing Dashboards collect KPIs and critical metrics

and display that data in a meaningful way. These Dashboards can also cull data from cloud services [11];

- **Executive Dashboards** - While not as in-depth as other options listed, executive Dashboards are full of vital real-time data that provide executives an overview of their company's performance. Typically, executive Dashboards consist of data from accounting software, website analytics, and Customer Relationship Management (CRM) [11];
- **Real-Time Dashboards** - By their very definition, Dashboards provide real-time data. Reports, conversely, offer information based on historical data. Real-time Dashboards are instantly updated so you can see how something is performing at that very moment. A visual display of the most important information needed to achieve one or more objectives, consolidated and arranged on a single screen so the information can be monitored at a glance [11].

2.2.2. Dashboard Platforms

There are some Data visualization platforms available, that can automatically create visualizations, enable you to create your own, or offer both capabilities. Next, are the most popular platforms.

- **Zoho Reports** - Zoho.com is a provider of award-winning online business apps. The Zoho Applications' suite lives in the cloud, so businesses can access their data wherever needed. A Dashboard is an effective way of organizing reports into a single page to have a quick insight into the Key Metrics at a glance. Zoho Reports provides a simple & intuitive drag and drop interface for creating Dashboards [12].
- **Sisense** – Sisense it's an intelligence software that provides analytic solutions and market insights for small t-o enterprise-level businesses. Sisense is one of only a few fully-functioning BI software systems that let non-technologically inclined users combine multiple data sets, customize Dashboards, generate data visualizations, and share them with other users [13].

- **Tableau** - Dashboarding with Tableau allows even non-technical users to create interactive, real-time visualizations in minutes. Sharing a Dashboards requires no programming, whether it's on Tableau Server, Tableau Online, or any portal or web page [14].
- **Google Analytics** - The Google Dashboard is organized according to the products you use, showing data associated with that product or service as well as direct links to that product's privacy and security settings. The Google Sites Dashboard is part of the Google Apps offering [15].
- **SAS Visual Analytics** - SAS Visual analytics is a form of inquiry in which data that provides insight into solving a problem is displayed in an interactive, graphical manner. SAS Visual Analytics provides a complete platform for analytics visualization, enabling to identify patterns and relationships in data that weren't initially evident. Interactive, self-service BI and reporting capabilities are combined with out-of-the-box advanced analytics so everyone can discover insights from any size and type of data, including text [16].

Next, a comparative table was made showing the main characteristics of the mentioned platforms (see Table 1).

	Zoho Reports	Sisense	Tableau	Google Analytics	SAS Visual Analytics
OS	Windows iOS Web-Based Android Windows Mobile	Windows Android iOS Web based	Windows Android iOS Web based	Windows Linux iOS Web-Based	Windows Android iOS

Clients	Worldwide Express, CH2MHill, Outdoor Sports Marketing	NASDAQ, Merck, Dannon, Booking, Comcast, NASA, ESPN, Sony	Accenture, Coco Cola, Bank of America, PayPal, Google, Skype, Citigroup, Dell, Walmart	Panasonic, Progressive, TransUnion	Staples, Scotia Bank, Australian Institute of Health and welfare
Price	50\$	By quote	35\$	Free	Price is based on server and user capacity
Cloud Platform	Yes	Yes	Yes	Yes	Yes
Free Versión Available	Yes	No	No	Yes	Yea

Table 1 - Dashboard Platforms Comparison.

All of the above platforms, summarized in Table 1, will serve as support to achieve this research. The work will evolve from an already designed and established platform that collects an extraordinary amount of data. The data used in this case study is related to surveys. The goal is to produce a dynamic Dashboard based on real-time / live information. Thus, Dashboard users will be able to interact with the information, based on an initial set of clues, charts, tables and reports, produced by the Dashboard itself. This will allow testing an existing set of data visualization techniques and tools and discover a new set of data visualization techniques and platforms.

The platforms mentioned above have a high cost for each of its services, and are very complex to understand, for this reason it is necessary to develop a Dashboard that meets the characteristics of these great platforms but which is free and easy to use by users. Thus, this is the starting point for this research to be successful in the academic and business area, in addition it is going to be our goal to make a web application responsive. Responsive web design is suitable for informational web pages, with information on the right side of the screen in desktop applications brought to the bottom of the page in mobile applications.

2.2.3. Data Visualization Techniques

There are also data visualization techniques, which play a very important role in organizations and allow through the use of images and graphics display data to make them more understandable, attractive, manageable and useful [17].

Visualizing Semi structured and Unstructured Data Using Word Clouds and Network Diagrams.

The variety of data brings challenges because semi structured, and unstructured data require new visualization techniques. A word cloud visual (where the size of the word represents its frequency within a body of text) can be used on unstructured data as a way to display high- or low-frequency words (see Figure 4) [18].

SAS Visual Analytics takes the concept of word clouds a step further by taking advantage of taxonomies and ontologies to make associations. Words are then organized into topics based on how the words are used. SAS Visual Analytics word clouds can display the hot topics of the day gleaned from such text analysis. Users can drill down by clicking on an individual topic to see exactly what words or phrases comprise that topic [18].

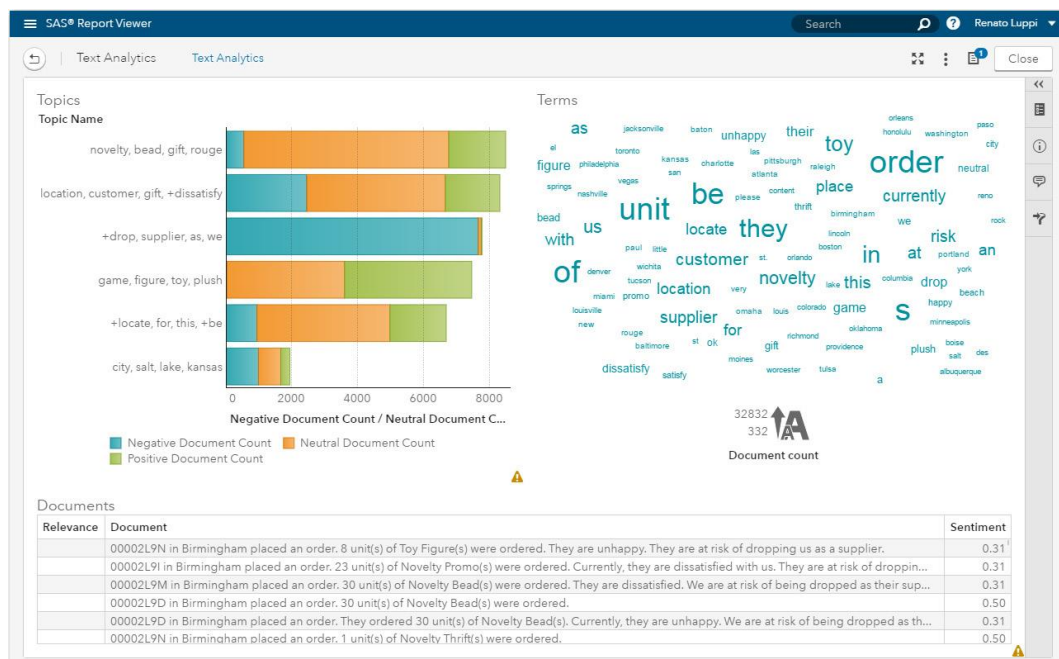


Figure 4 - A word cloud shows the words or phrases associated with a topic.

Another visualization technique that can be used for semi structured or unstructured data is the network diagram. Network diagrams view relationships in terms of nodes (representing individual actors within the network) and ties (which represent relationships between the individuals, such as friendship, kinship, organizations, business relationships, etc.). These networks are often depicted in a diagram where nodes are represented as points and ties are represented as lines [18].

Network diagrams can be used in many applications and disciplines. For example, businesses analyze social networks to understand their interactions with customers, while counterintelligence and law enforcement might map a clandestine or covert organization such as an espionage ring, an organized crime family or a street gang. You can also superimpose the network diagram on a map, for example, to show the relationship or product sales across geographic areas (see Figure 5) [18].

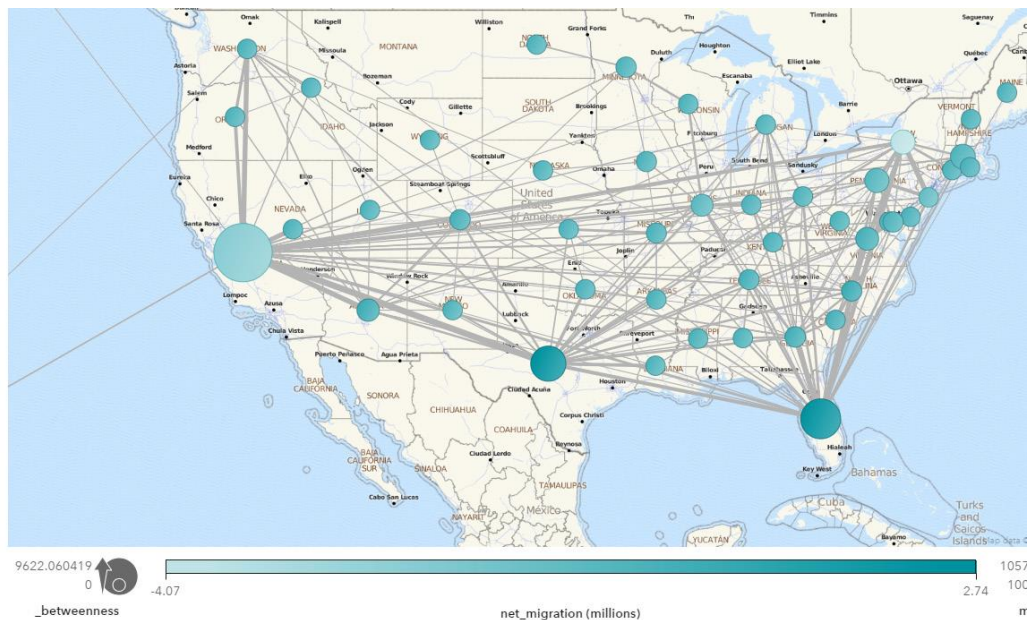


Figure 5 - Network diagrams explore relationships within a data set, including connections across geographic areas.

Visualization with Correlation Matrices

A correlation matrix shows a table which lists the correlation coefficients of the columns and rows and combines big data and fast response times to quickly identify which variables are related. It also shows how strong the relationships are between variables [19].

SAS Visual Analytics makes it easy to assess the relationships. Simply select a group of variables and drop them into a visualization pane. The intelligent autocharting function displays a color-coded correlation matrix that quickly identifies strong and weak

relationships between the variables [19] Darker boxes indicate a stronger correlation; lighter boxes indicate a weaker correlation (see Figure 6).

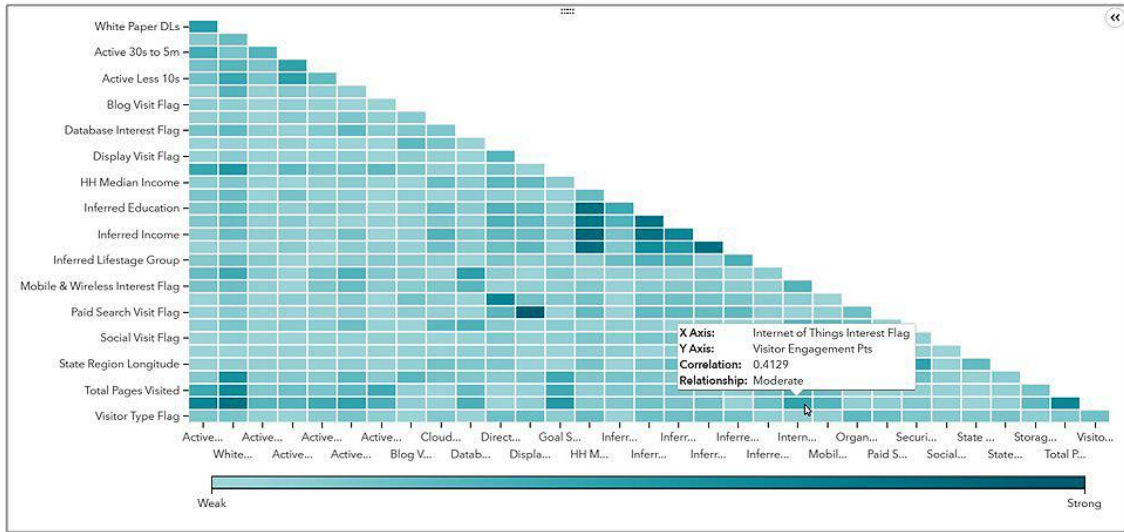


Figure 6 - In this correlation matrix, darker boxes indicate a stronger correlation; lighter boxes indicate a weaker correlation.

Data Visualization Made Easy with Autocharting

In SAS Visual Analytics, intelligent autocharting produces the best visual based on what data you drag and drop onto the visual palette. It is important to note that autocharting may not always create the exact visualization user had in mind. In that case, you also can select a specific visual to build. However, when you are first exploring a new data set, autocharts are useful because they provide a quick view of the data. For example, with autocharting, when a single measure is selected, distribution of that measure is shown [20] (see Figure 7).

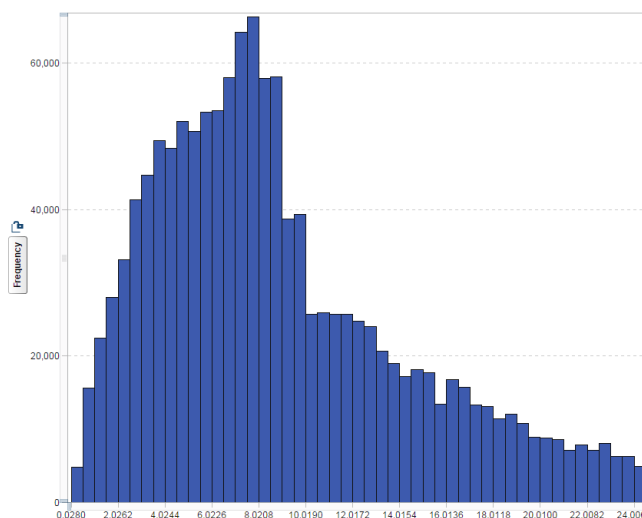


Figure 7 - Autocharting in SAS Visual Analytics produces a bar chart to show the distribution of a single measure.

Visualizing Flow with Sankey Diagrams

Sankey diagrams use path analysis to show the dynamics of how transactions move through a system (e.g., how customers navigate on a website). The diagrams display a series of linked nodes, where the width of each link indicates the frequency of the link or the value of a measure. This enables you to see flow patterns and recognize trends such as where customers enter your website, where they navigate to and where they exit. It is possible to identify successful flow patterns or isolate flows that failed to deliver the desired action (see Figure 8) [21].

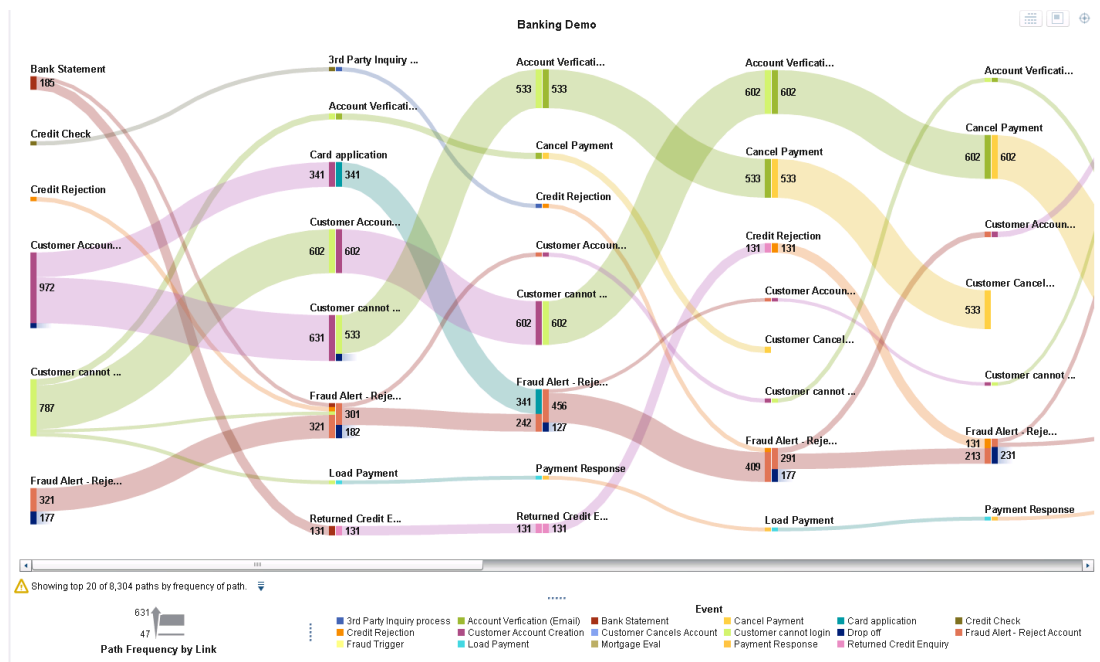


Figure 8 - A Sankey diagram displays a series of linked nodes, where the width of each node indicates the frequency of the link or value of the measure.

Visualization for Mobile Devices

Proliferation of mobile devices means that businesses need to deliver company information to smartphones and tablets any time and from anywhere [22].

In mobile devices is necessary to consider:

- **Fluid grids:** The grid is the tool that designers use to lay out their designs, regardless of whether those designs will be viewed online or offline. In the following figure, a grid design has been transformed for three screen shapes (see Figure 9);

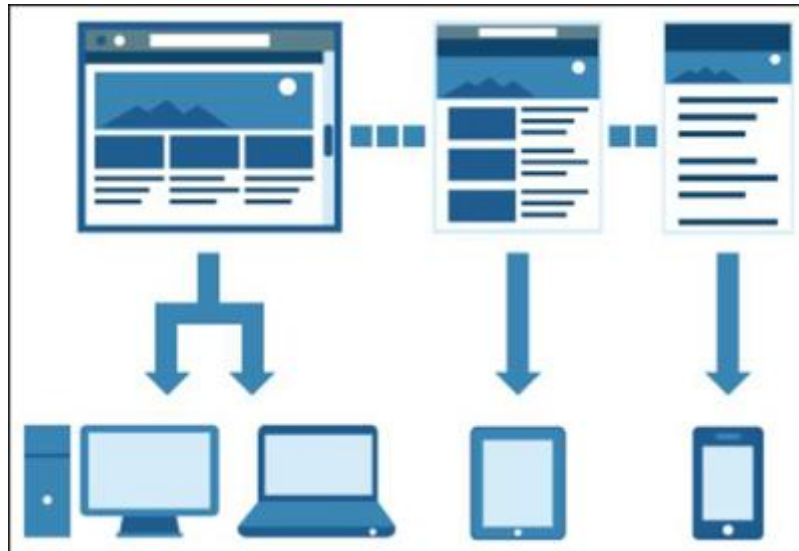


Figure 9 - Grid design translated to different screens.

- **Flexible images:** When you have images in a blog post or on your website, you want to ensure that they're not cut off or distorted. Making sure that your images can reformat themselves is important;
- **Media queries:** When you enter a search term in a search engine such as Google or Bing, you want to be able to view the results on any of your screens. The content needs to be viewed on any mobile device a user might have. What makes viewing it on a variety of devices a challenge is that the designer has no way to know what the returned result will be.

This page was intentionally left blank

3. Project Outline

In this chapter we describe the Case Studies, the data model, and the technologies and tools used in the development of the Dashboard.

3.1. Case Studies

This chapter will present the generic case studies considered in the implementation of the project. These were the basis of the analysis and specification of the data model to support the system to be implemented.

Pedagogical questionnaires

The pedagogical questionnaires elaborated by the pedagogical councils of the 5 schools of the Polytechnic of Leiria, where the student questionnaires are composed of 15 questions, 8 questions for assessing the functioning of the curricular unit and 7 questions for the evaluation of pedagogical performance (see Figure 10) were used as the first data source for the case study.

POLITÉCNICO DE LEIRIA

(ESTG) ... Estudantes - 2º Sem 2016/2017
Curso: Licenciatura em Engenharia Informática [PL]
Código: 985238 TESTIGPL_S2
Unidade curricular: Laboratório de Tecnologias de Informação

Avaliação do funcionamento da Unidade Curricular (UC).
Assessment of the Curricular Unit (UC).

	1	2	3	4	5	NA
Posicionamento da UC no plano de estudos. Adequacy of UC position in the study plan.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Adequação do volume de trabalho ao número de ECTS. Adjustment between workload and ECTS (credits).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Recursos físicos (salas de aula, laboratórios, oficinas). Equipment and physical facilities (classrooms, laboratories, ateliers).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Adequação do número de alunos em aula. Adequacy of the number of students per class.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Articulação das várias componentes da UC (teórica e prática). Coordination of the different components of the UC (theoretical and practical).	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Adequação dos materiais e bibliografia. Adequacy of study materials and bibliography.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Adequação dos métodos e critérios de avaliação. Adequacy of assessment methods and criteria.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Apreciação global do funcionamento da UC. Overall assessment of the UC.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Comentários
Comments

Anterior Seguinte

Figure 10 - Pedagogical questionnaires.

PPE-15

The partial questionnaire on patient satisfaction of health units developed by the Picker Institute (see Figure 11), composed of 15 questions, one of them with an alternative path, that is, whether the following question or questions depend on the subsequent response.

<p>8. If you had any anxieties or fears about your condition or treatment, did a nurse discuss them with you?</p> <p>Yes, completely/Yes, to some extent/No/I didn't have any anxieties or fears</p> <p>9. Did you find someone on the hospital staff to talk to about your concerns?</p> <p>Yes, definitely/Yes, to some extent/No/I had no concerns</p> <p>10. Were you ever in pain?</p> <p>Yes/No</p> <p>If yes...</p> <p>Do you think the hospital staff did everything they could to help control your pain?</p> <p>Yes, definitely/Yes, to some extent/No</p> <p>11. If your family or someone else close to you wanted to talk to a doctor, did they have enough opportunity to do so?</p> <p>Yes, definitely/Yes, to some extent/No/No family or friends were involved/My family didn't want or need information/I didn't want my family or friends to talk to a doctor</p> <p>12. Did the doctors or nurses give your family or someone close to you all the information they needed to help you recover?</p> <p>Yes, definitely/Yes, to some extent/No/No family or friends were involved/My family or friends didn't want or need information</p> <p>13. Did a member of staff explain the purpose of the medicines you were to take at home in a way you could understand?</p>

Figure 11 - Questionnaire on patient satisfaction.

Requirements Identification

The questions of these questionnaires have many types of possible answers, such as [23]:

- **Open-ended Questions** - An open-ended question cannot be answered with a "yes" or "no" response, or with a static response. Open-ended questions are phrased as a statement which requires a response;
- **Closed-ended Questions** - Four types of closed-ended questions are most commonly used: rating scale, forced choice, dichotomous and demographic/firmographic questions;
- **Rating scale questions** - Respondents assess the issue based on a given dimension. Two frequently used types of rating scale questions are Likert-type scales and semantic differential scales;
- **Semantic differential** - In a semantic differential scale, each end of the scale marked is with different or opposing statements;
- **Multiple Choice Questions** - Multiple choice questions ask the respondent to choose between two or more answer options. Questions can be as simple as "yes/no" or can give a choice of multiple answers.

3.2. Data Model

In this chapter we explain the logic of the data model used by the application (see Figure 12)

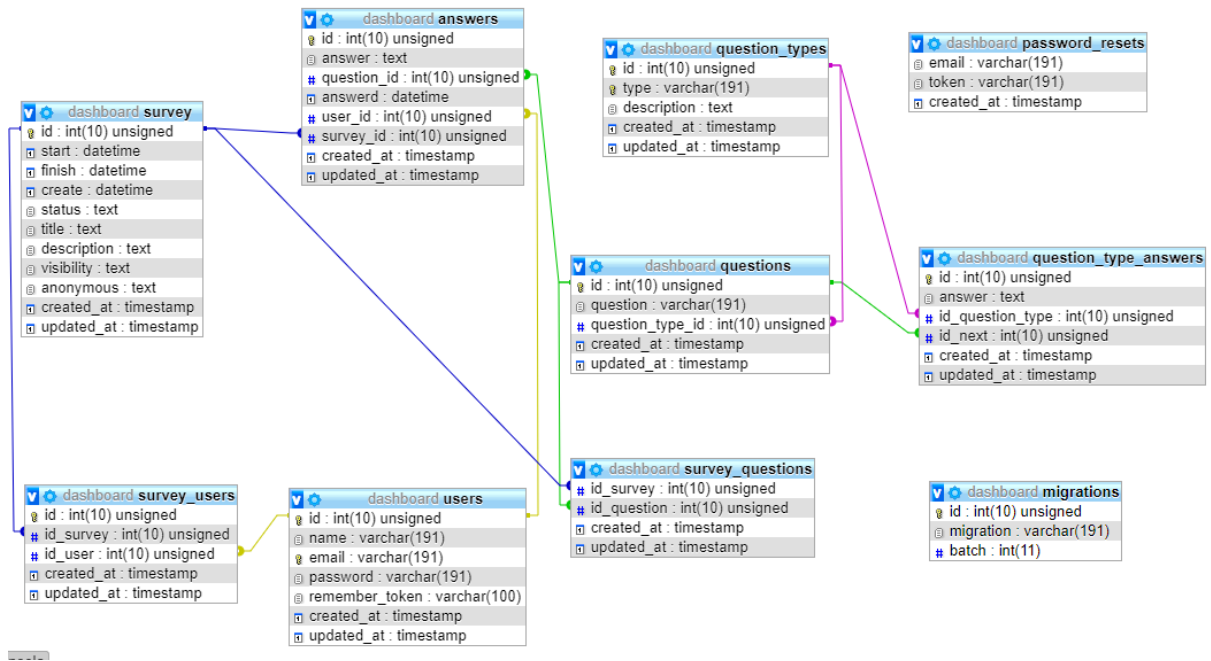


Figure 12 – Generic Data Model.

The model and the database were the result of another research, responsible for the collection of information from the surveys [24]. It has a generic structured in order to be able to accommodate other type of surveys in the future. The aim of this model is to support information related to surveys, mainly, considering the need to represent questions, answers, type of answer, etc.

The purpose of this research is to graphically represent through a dynamic Dashboard all the information related to the answers given by the users that fulfill the surveys implemented and made available in the platform. As mentioned previously, one important aspect of the data model it is the entire structure of the database that has been designed with the aim of being the most generic possible, so that all questionnaires containing open-ended questions or multiple types of multiple choice can be entered into the database, and the developed application. Throughout the structure care is taken to insert extra attributes especially for statistics, such as in the Survey entity, the start, created, and finish attributes, which contain the start, create, and end dates.

3.3. Technologies and tools

This project focuses on making a web Dashboard, for which we mention the tools that are used in the development (see Figure 13).

To join the MySQL database in Laravel, the migration process in the tables is carried

out, defining all the fields and relationships of the tables. Then are performed the controllers, where the functions required by the Dashboard are accomplished. In addition to this, are created access routes for the views. Laravel contains all web development, PHP, HTML, Javascript, CSS in a MVC Model.

The real-time communication of the Dashboard is accomplished by using a Redis server that is responsible for “notify” when new data is entered into the database which allows the updating of the Dashboard information and graphics.

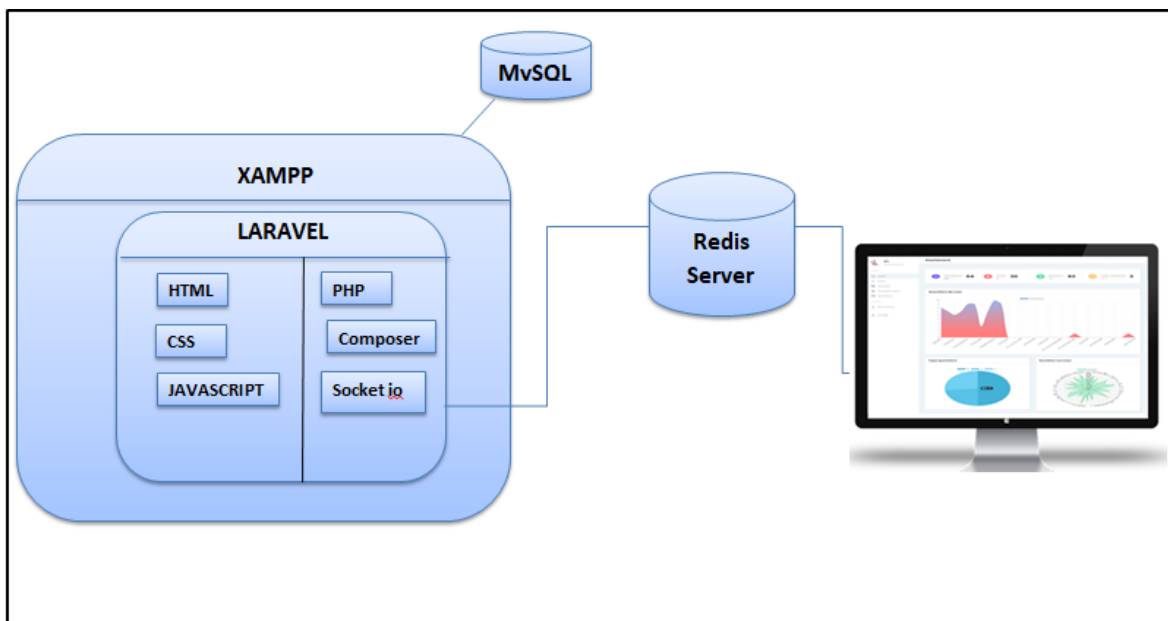


Figure 13 – General Architecture with the technology used.

Laravel

Laravel [25] is a web application framework with expressive, elegant syntax. Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching.

The Laravel framework has a few system requirements. Of course, all of these requirements are satisfied by the Laravel Homestead virtual machine, so it's highly recommended that you use Homestead as your local Laravel development environment [26].

However, if not using Homestead, need to make sure your server meets the following requirements: PHP \geq 7.1.3, OpenSSL PHP Extension PDO PHP Extension, Mbstring PHP Extension, Tokenizer PHP Extension, XML PHP Extension, Ctype PHP Extension, JSON PHP Extension [26].

MySQL

MySQL [27] is a database management system. A database is a structured collection of data. To add, access, and process data stored in a computer database, a database management system such as MySQL Server was needed.

PHP

PHP [28] is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

Artisan Console

Artisan [29] is the command-line interface included with Laravel. It provides a number of helpful commands that can assist you while you build your application. To view a list of all available Artisan commands, you may use the list command: *php artisan list*, the commands are available in Appendix B.

Composer

Composer [28] is a tool for dependency management in PHP. It allow to declare the libraries in the project depends on and it will manage (install/update) . Composer is not a package manager in the same sense as Yum or Apt are[30]. Yes, it deals with "packages" or libraries, but it manages them on a per-project basis, installing them in a directory (e.g. vendor) inside the project. By default, it does not install anything globally. Thus, it is a dependency manager. It does however support a "global" project for convenience via the global command.

Javascript Libraries

Javascript libraries [31] will help to create and customizable charts for the projects. These libraries let present complex statistics quickly and effectively such as Chart.js, Highcharts.js, bootstrap.js, jquery.min.js.

Chart JS

Chart.js [32] is a JavaScript library which provides a toolkit for data visualization in web browsers. This means it is essentially a collection of prewritten JavaScript code. The chart library was built to make working with HTML, CSS and SVG in conjunction with data easier.

Highcharts JS

Highcharts [33] is a charting library written in pure JavaScript, offering an easy way of adding interactive charts to your web site or web application. Highcharts currently supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, arearange, areasplinerange, columnrange, bubble, box plot, error bars, funnel, waterfall and polar chart types.

Bootstrap

Bootstrap [34] is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build the entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

Jquery.min.js

jQuery [35] is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

Jquery – Datatable

DataTables is a plug-in for the jQuery Javascript library. It is a highly flexible tool, build upon the foundations of progressive enhancement, that adds all of these advanced features to any HTML table [36].

Carbon API

Carbon [28] is a simple PHP API extension for DateTime, the Carbon class is inherited from the PHP DateTime class. Carbon has all of the functions inherited from the base DateTime class. This approach allows you to access the base functionality if you see anything missing in Carbon but is there in DateTime.

Redis

Redis [37] is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps and geospatial indexes with radius queries. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster. Redis also supports trivial-to-setup master-slave asynchronous replication, with very fast non-blocking first synchronization, auto-reconnection with partial resynchronization on net split.

Node.js

Node.js [38] is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, thus, perfect for data-intensive real-time applications that run across distributed devices.

Socket. IO

Socket.IO [39] is a JavaScript library for real-time web applications that enables real-time, bi-directional communication between web clients and servers. Socket.IO has two components: a client-side library that runs in the browser, and a server-side library for Node.js. Both components have nearly identical APIs.

Font Awesome

Font Awesome [40] is a font and icon toolkit based on CSS and LESS. In the Dashboard was used in the tables, in the search, ordering and pagination.

Responsive Web Design

Responsive Web Design [41] is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones).

Operating System

The tools and technologies used in this research are multiplatform, which guarantees that it works correctly in Windows, Linux and Mac OS.

XAMPP

XAMPP [42] is a completely free, easy to install Apache distribution containing MySQL and MariaDB [27], PHP [43], and Perl [44]. XAMPP provides a free all-in-one tool to install Drupal [45], Joomla [46], WordPress [47], Laravel [48], Codeigniter [49] and many other popular open source apps on top of XAMPP. Also provides an ideal local development environment but is not meant for production deployments.

Sublime Text

Sublime Text [28] is a proprietary cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses. The packages are available in Appendix C.

Git

Git [50] is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance. The commands are available in Appendix D.

Bitbucket

Bitbucket [51] is a web-based version control repository hosting service owned by Atlassian, for source code and development projects that use either Mercurial or Git revision control systems. Bitbucket offers both commercial plans and free accounts. Bitbucket is more than just Git code management. Bitbucket gives teams one place to plan projects, collaborate on code, test and deploy. The source code is hosted in bitbucket.

Git-Hub

GitHub [52] is a development platform inspired by the way the work. From open source to business, can host and review code, manage projects, and build software alongside millions of other developers. The Documents are hosted in Git-hub

This page was intentionally left blank

4. Dashboard Implementation

In this chapter we describe the Dashboard development process, some security considerations in the Dashboard. Details about the Dashboard sections and web hosting are also presented.

4.1. Dashboard Sections and Features

Once the Dashboard is available, the sections are identified and are detailed below:

Menu and Statistics sections

This section shows the option menu and its content and details statistical data of the surveys such as number of questions, number of answers, number of users, type of questions, number of active and inactive surveys, the user with the most answers. (see Figure 14).

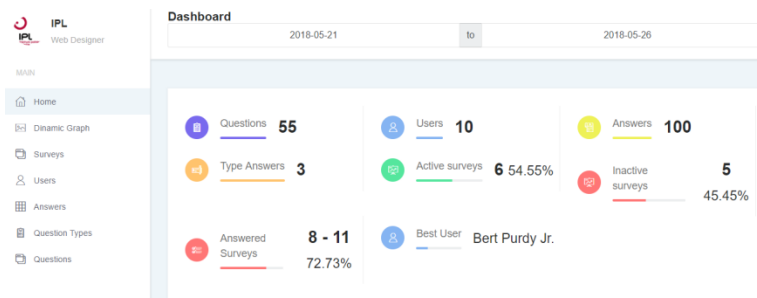


Figure 14 – Menu and Statistics view sections.

Temporal Analysis

The Dashboard allows the user to navigate between dates, which allows a temporary analysis of the information depending on the selected dates. All graphics are updated through the selected dates (see Figure 15).

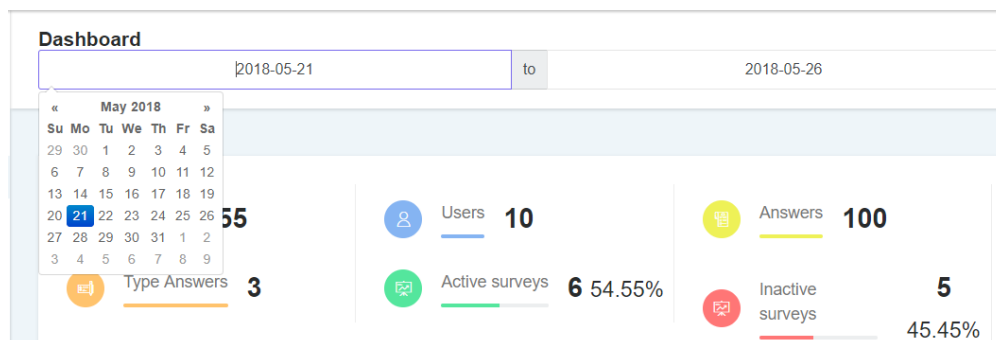


Figure 15 - Temporal Analysis.

Home Section

This section contains the statistics, the temporal analysis and the visualization through charts of: Question by user, Types of question, Surveys Duration, Surveys count and Types of surveys (see Figure 16).

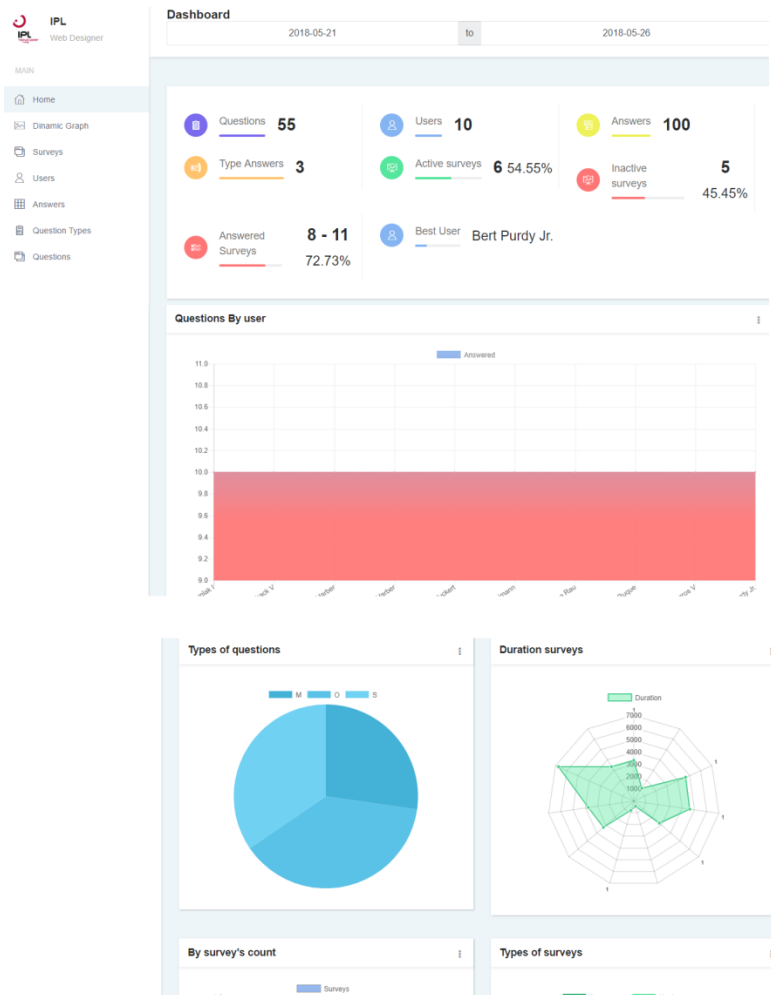


Figure 16 - Home Section.

Dynamic Chart Section

The Dashboard allows the user to select what to represent in the graphics: Question by user, Surveys by user, Answer by user, Question by types and allows the user to select the chart to graph: Line chart, Doughnut Chart, Pie Chart (see Figure 17).

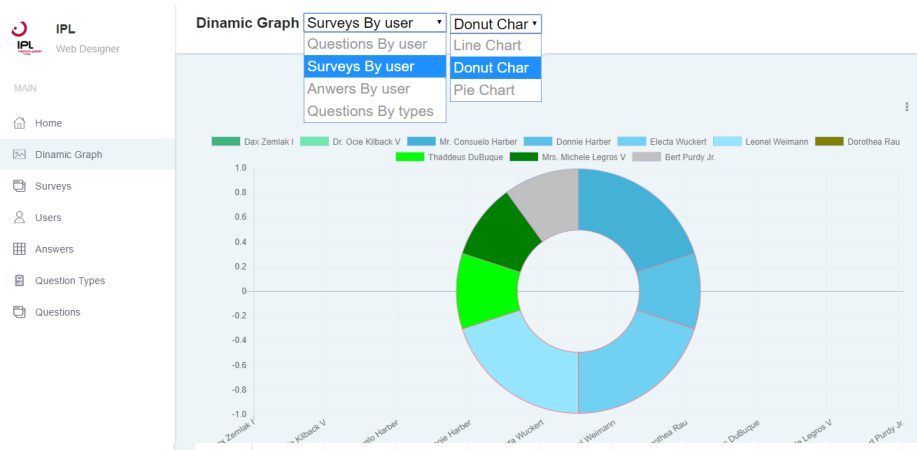


Figure 17 - Dynamic Chart section.

Surveys Section

This section contains the list of all the surveys, this can be ordered by each field of the table, this can also be filtered by dates the information of the list of surveys (see Figure 18).

#	Start	Finish	Duration	Title	Questions	Users	Status in this page
1	2018-05-23 16:31:16	2018-05-25 23:44:56	2	Enim eum quas nostrum ut sunt consequat. Nihil fugiat doloreque fugiat error esse doloremque. Dolor iusto sed provident sed error id. Ullam et vero autem quisquam blanditibus ullam.	5 per survey	0	Active
2	2018-05-25 17:58:00	2018-05-26 15:10:20	0	Earum aut asperiores quas sint qui in dolor. Tempore voluptas ullam aliquid. Saepe cumque voluptas esse. Quo quis minus reprehenderit. Quaeerat nam inventore quasi dolores omnis alias voluptatem.	5 per survey	0	Inactive
3	2018-05-22 10:40:47	2018-05-25 16:10:49	3	Labore enim magni sit qui at autem atque. Accusantium reprehenderit beatae maxime. Rerum aliquam quod rerum ipsum ea incidunt nihil. Sed ut ad sunt earum provident eveniet aliquid.	5 per survey	2	Inactive
4	2018-05-22 14:39:56	2018-05-25 19:28:36	3	Quis in vitae perferendis blanditibus cumque rem. Alias nesciunt ut quia adipisci blanditibus. Est qui officis ad velit et. Iure non rerum eos eveniet nempe.	5 per survey	1	Inactive

Figure 18 - Surveys section.

In the table can access the surveys by clicking on them in the title field. There are two types of surveys that are text type and numeric type. In the Text Surveys the content of the survey was shown (see Figure 19).

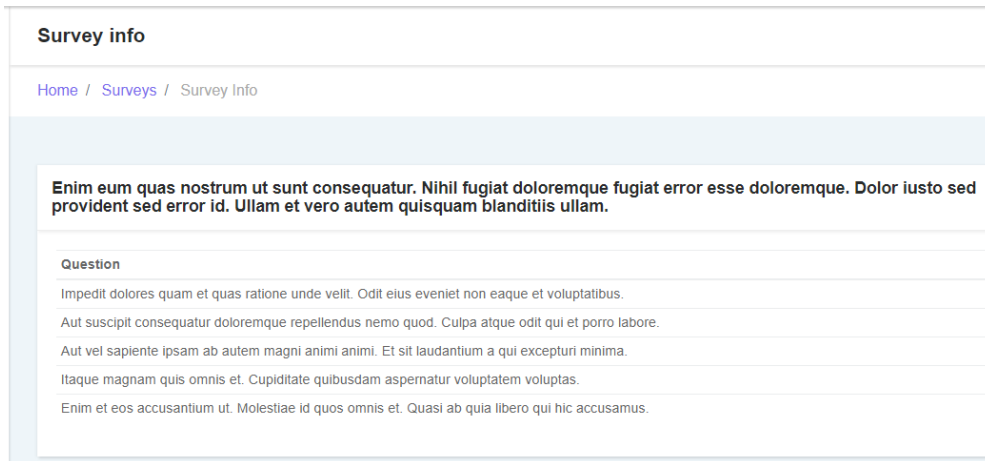


Figure 19 - Text Survey.

In the Numeric Surveys the content of the survey and a statistical analysis of the answers are shown. Figure 20 presents the answers of the Numeric Survey.

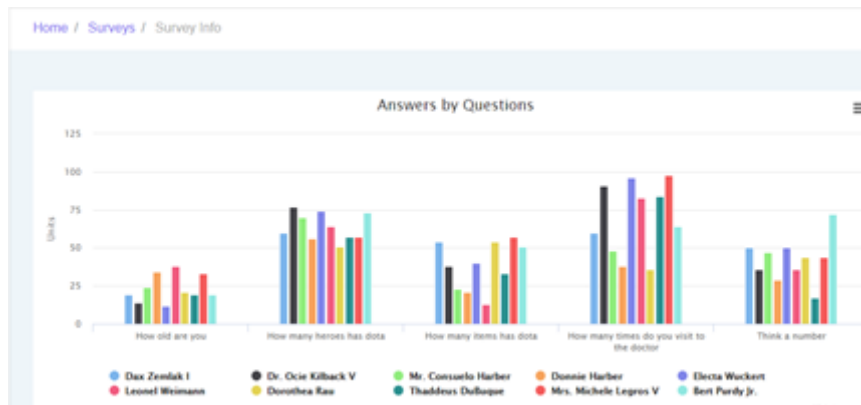


Figure 20 – Answers of Numeric Survey.

Figure 21 presents the statistics in a chart of the answers of the Numeric Survey.

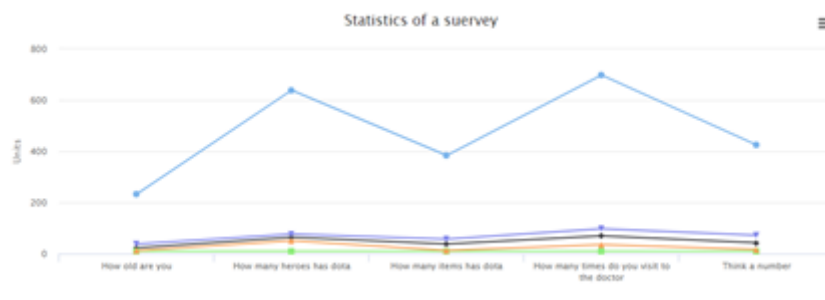


Figure 21 - Statistics of the Answers of Numeric Survey in a chart.

Figure 22 presents the statistics in a table of the answers of the Numeric Survey.

Highcharts.com

numeric survey numeric

Question	Total	Media	Answers	Min	Max
How old are you	233	23.3	10	12	38
How many heroes has dota	629	63.9	10	51	77
How many items has dota	384	38.4	10	13	57
How many times do you visit to the doctor	698	69.8	10	26	98
Think a number	426	42.6	10	17	72

Figure 22 - Statistics of the Answers of Numeric Survey in a table.

Users Section

This section shows the list of available users and the number of surveys that have answered (see Figure 23).

Dynamic Graph
Surveys
Users
Answers
Question Types
Questions

User List

Show 10 entries Search:

#	name	email	Answered Surveys
1	Dax Zemlak I	jensen87@example.org	0 - 11
2	Dr. Ocie Kilback V	alessia.beier@example.org	0 - 11
3	Mr. Consuelo Harber	lyla42@example.org	2 - 11
4	Donnie Harber	gstehr@example.org	1 - 11
5	Electa Wuckert	leffler.kay@example.com	2 - 11
6	Leonel Weimann	roscoe.reichert@example.com	2 - 11
7	Dorothea Rau	georgianna.simonis@example.net	0 - 11
8	Thaddeus DuBuque	katrine.watsh@example.org	1 - 11
9	Mrs. Michele Legros V	grath@example.net	1 - 11
10	Bert Purdy Jr.	crooks.annamarie@example.com	1 - 11

Figure 23 - Users Section.

Answers Section

This section shows the list of answers, the date, the survey to which it belongs and the user who answered (see Figure 24).

Home
Dynamic Graph
Surveys
Users
Answers
Question Types
Questions

Answers List

Show 10 entries Search:

#	Answer	Answered	Survey	User
1	BWGTyqQsw9	2018-05-26 03:31:52 3 weeks ago	9	Dax Zemlak I
2	lh1ENabnU8	2018-05-26 03:31:52 3 weeks ago	9	Dax Zemlak I
3	gRRth2nq5M	2018-05-26 03:31:52 3 weeks ago	9	Dax Zemlak I
4	gHJWAhnqKU	2018-05-26 03:31:52 3 weeks ago	9	Dax Zemlak I
5	zsbB6ZMhmH	2018-05-26 03:31:53 3 weeks ago	9	Dax Zemlak I

Figure 24 - Answers Section.

Question Types Section

This section shows the list of types of questions that exist, detailing their identification, type and description (see Figure 25).

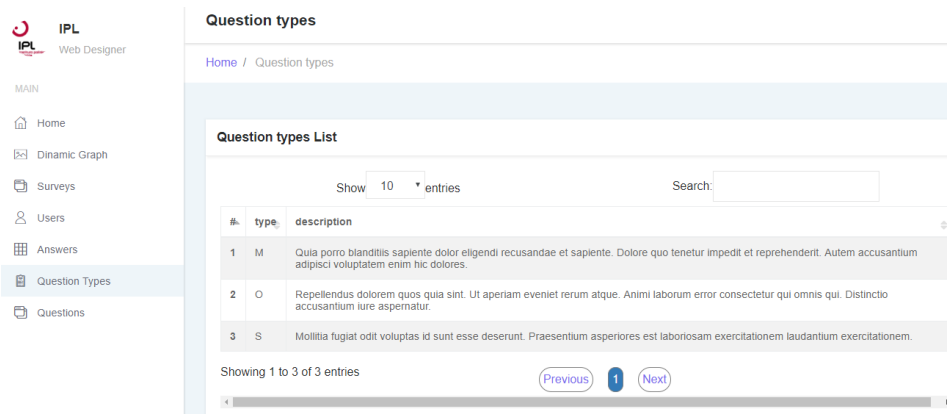


Figure 25 - Question Types Section.

Questions Section

This section shows the list of questions and to what type they belong (see Figure 26).

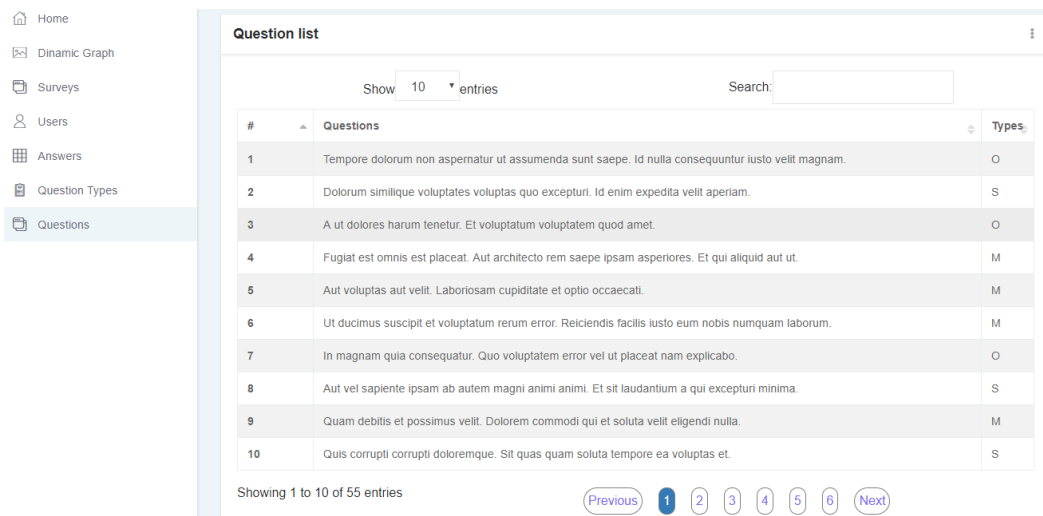


Figure 26 - Questions Section.

4.2. Implementation Details

Once an initial prototype of a dashboard has been defined (see Appendix E).

After that itit was necessary to prepare the work environment (see Appendix F).

First, the creation of the database in Laravel through the migration process and seeders, to the development process it was used models, controllers, events views, which are detailed below:

4.2.1. Migrations

Migrations are like version control for the database, allowing the team to easily modify and share the application's database schema. Migrations are typically paired with Laravel's schema builder to easily build the application's database schema [53].

In the console were used the next commands to create the tables used:

```
php artisan make:migration create_users_table  
php artisan make:migration create_questions_table  
php artisan make:migration create_question_types_table  
php artisan make:migration create_survey_table  
php artisan make:migration create_answers_table  
php artisan make:migration create_survey_questions_table  
php artisan make:migration create_survey_users_table  
php artisan make:migration create_questions_table  
php artisan make:migration create_question_type_answers_table  
php artisan make:migration
```

Migration Structure - The migration class contains two methods: up and down. The up method is used to add new tables, columns, or indexes to your database, while the down method should simply reverse the operations performed by the up method (see Figure 27).

```

1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateUsersTable extends Migration
8  {
9      /**
10     * Run the migrations.
11     *
12     * @return void
13     */
14     public function up()
15     {
16         Schema::create('users', function (Blueprint $table) {
17             $table->increments('id');
18             $table->string('name');
19             $table->string('email')->unique();
20             $table->string('password');
21             $table->rememberToken();
22             $table->timestamps();
23         });
24     }
25
26     /**
27     * Reverse the migrations.
28     *
29     * @return void
30     */
31     public function down()
32     {
33         Schema::dropIfExists('users');
34     }
35 }

```

Figure 27 - Migration structure, Table users.

All tables contain this structure, but the content depends on the fields that each table has.

4.2.2. Seeders

Laravel includes a simple method of seeding the database with test data using seed classes. All seed classes are stored in the *database/seeds* directory. Seed classes may have any name you wish, but probably should follow some sensible convention, such as *UsersTableSeeder*, etc [54].

In the console the next commands were used:

```

php artisan make:seeder UserSeeder
php artisan make:seeder AnswerSeeder
php artisan make:seeder QuestionSeeder
php artisan make:seeder QuestionTypeAnswerSeeder
php artisan make:seeder QuestionTypeSeeder
php artisan make:seeder SurveyQuestionSeeder
php artisan make:seeder SurveySeeder
php artisan make:seeder SurveyUserSeeder
php artisan db:seed

```

Seeders Structure - The class contains the run method, in this method the number of fields of the table is defined (see **Figure 28**).

```

<?php

use Illuminate\Database\Seeder;

use App\User;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        factory(User::class,10)->create([]);
    }
}

```

Figure 28 - Seeder Structure, table user.

4.2.3. Model factory

In the directory database/factories is created a file named: *ModelFactory.php*, in this file it will work. It is possible to use model factories to conveniently generate large amounts of database records (see Figure 29).

```

1 <?php
2
3 /*
4 -----
5 Model Factories
6 -----
7
8 Here you may define all of your model factories. Model factories give
9 you a convenient way to create models for testing and seeding your
10 database. Just tell the factory how a default model should look.
11
12 */
13
14 /** @var \Illuminate\Database\Eloquent\Factory $factory */
15 $factory->define(App\User::class, function (Faker\Generator $faker) {
16     static $password;
17
18     return [
19         'name' => $faker->name,
20         'email' => $faker->unique()->safeEmail,
21         // 'password' => $password ?: $password = bcrypt('secret'),
22         'password' => $password ? $password = bcrypt('12345'),
23         'remember_token' => str_random(10),
24     ];
25 });
26
27 $factory->define(App\Question_type::class, function (Faker\Generator $faker) {
28
29     return [
30         'type' => $faker->name,
31         'description' => $faker->text(),
32     ];
33 });
34
35 $factory->define(App\Question::class, function (Faker\Generator $faker) {
36     $questions = \DB::table('question_types')->select('id')->get()->toArray();
37
38     return [
39         'question' => $faker->text(100),
40         'question_type_id' => $faker->randomElement($questions)->id,
41     ];
42 });
43

```

Figure 29 - Model factory of the tables.

In the file *DatabaseSeeder* located in the directory database/seeds is called the seed classes created (see Figure 30).

```

<?php
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $this->call(UserSeeder::class);
        $this->call(QuestionTypeSeeder::class);
        $this->call(QuestionSeeder::class);
        $this->call(SurveySeeder::class);
        $this->call(SurveyQuestionSeeder::class);
        $this->call(AnswerSeeder::class);
        $this->call(QuestionTypeAnswerSeeder::class);
        $this->call(SurveyUserSeeder::class);
    }
}

```

Figure 30 - Seeders Call

Figure 31 shows how the data is created in the tables of the database, these data respect the relationships between the tables.

	id	question	question_type_id	created_at	updated_at
<input type="checkbox"/> Editar Copiar Borrar	1	Dolore minus rerum magni est id minima quo. Placea... Longitud original 98	3	2017-12-12 00:52:45	2017-12-12 00:52:45
<input type="checkbox"/> Editar Copiar Borrar	2	Accusantium corrupti doloribus aut non est aut mol...	1	2017-12-12 00:52:45	2017-12-12 00:52:45
<input type="checkbox"/> Editar Copiar Borrar	3	Quam consequatur numquam quod nostrum consequatur...	2	2017-12-12 00:52:45	2017-12-12 00:52:45
<input type="checkbox"/> Editar Copiar Borrar	4	In quia et consectetur voluptatem harum. Sunt dele...	3	2017-12-12 00:52:45	2017-12-12 00:52:45
<input type="checkbox"/> Editar Copiar Borrar	5	Ad et temporibus ea asperiores consequatur. Labori...	3	2017-12-12 00:52:45	2017-12-12 00:52:45
<input type="checkbox"/> Editar Copiar Borrar	6	Eaque voluptatem ut harum atque. Ea architecto per...	1	2017-12-12 00:52:45	2017-12-12 00:52:45
<input type="checkbox"/> Editar Copiar Borrar	7	Alias et similique animi enim ut. Amet in architec...	1	2017-12-12 00:52:45	2017-12-12 00:52:45
<input type="checkbox"/> Editar Copiar Borrar	8	Vitae ex porro ut iusto reprehenderit sapiente exc...	3	2017-12-12 00:52:46	2017-12-12 00:52:46
<input type="checkbox"/> Editar Copiar Borrar	9	Incidunt sint enim aut debitis. Tempore molestias ...	1	2017-12-12 00:52:46	2017-12-12 00:52:46
<input type="checkbox"/> Editar Copiar Borrar	10	Reiciendis distinctio aperiam eius quae quaerat un...	1	2017-12-12 00:52:46	2017-12-12 00:52:46
<input type="checkbox"/> Editar Copiar Borrar	11	Rerum perferendis corrupti illo dolor. Voluptatem ...	1	2017-12-12 00:52:46	2017-12-12 00:52:46
<input type="checkbox"/> Editar Copiar Borrar	12	Laborum voluptatum sapiente unde. Eos qui consequu...	3	2017-12-12 00:52:46	2017-12-12 00:52:46
<input type="checkbox"/> Editar Copiar Borrar	13	Labore qui enim quia enim. Magnam aliquid animi nu...	1	2017-12-12 00:52:46	2017-12-12 00:52:46
<input type="checkbox"/> Editar Copiar Borrar	14	Vel quod illum est. Ratione dolore qui nesciunt ve...	1	2017-12-12 00:52:46	2017-12-12 00:52:46
<input type="checkbox"/> Editar Copiar Borrar	15	Ipsum sint rem at rerum. Natus vel magni qui.	2	2017-12-12 00:52:46	2017-12-12 00:52:46

Figure 31 - Data created in the table.

4.2.4. Models

Laravel provides a simple ActiveRecord implementation for working with the database. Each database table has a corresponding "Model" which is used to interact with that table. Models allow you to query for data in your tables, as well as insert new records into the table. For to generate a database migration is necessary use the `-m` option [54].

To create the model, in the the next commands were executed:

```
php artisan make:model Answer -m
php artisan make:model Question -m
php artisan make:model Question_type -m
php artisan make:model Question_type_answers -m
php artisan make:model Survey -m
php artisan make:model Survey_questions -m
php artisan make:model Survey_user -m
```

Model Structure - In the model the name of the table and its fields were defined, and in a function the models of the tables with which it has a relationship were defined, the next figure shows an example (see Figure 32).

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Answer extends Model
{
    protected $table = "answers";

    protected $fillable = [
        'answer', 'question_id', 'answerd', 'user_id', 'survey_id'
    ];

    public function answer(){
        return $this->belogsto('App\Question');
        return $this->belogsto('App\User');
        return $this->belogsto('App\Survey');
    }
}
```

Figure 32 - Model Structure.

4.2.5. Controllers

Instead of defining all of the request handling logic as Closures in route files, it is possible to organize this behavior using Controller classes. Controllers can group related request handling logic into a single class. Controllers are stored in the app/Http/Controllers directory. The controllers have the main functions of the classes, such as queries to the database, control over the views to which the user will access [55].

In this case we have a controller that performs the functions of query, data insertion, depending on the case that is needed. And a general controller for the functions of the

Dashboard, such as real-time communication, connection to database queries to the database, to create the controllers, the next commands were executed in the console:

```
php artisan make:controller Answer
php artisan make:controller Dashboard
php artisan make:controller Question_types
php artisan make:controller Questions
php artisan make:controller Users
```

Controllers Structure - The structure of the controller will depend on the needs of the programmer, in the specific case of the image (see Figure 33), the functions perform paginations function, redirection to previously created views, and storage in the database from the view.

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Answer;
use App\Events\AnswerEvent;
class Answers extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(){
        $answers = Answer::paginate(10);
        return view('answers.list',['answers'=>$answers]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        $answer = factory(Answer::class)->make([]);
        $answer->save();
        event(new AnswerEvent($answer));
        return redirect()->route('answers.index');
    }
}
```

Figure 33 - Controllers Structure.

4.2.6. Events

Laravel's events provides a simple observer implementation, allowing to subscribe and listen for various events that occur in the application. Event classes are typically stored in the `app/Events` directory, while their listeners are stored in `app/Listeners`. Events serve as a great way to decouple various aspects of the application, since a single event

can have multiple listeners that do not depend on each other [56]. The dashboard implements events and listeners to real-time communication.

To create an event it was necessary add listeners and events to the EventServiceProvider we used the command:

```
php artisan event:generate
```

This command has generated any events or listeners that are listed in the EventServiceProvider (see Figure 34). Of course, events and listeners that already exist will be left untouched.

In this research it was created events and listener for: Answers, users, questions, questions_type and survey.

Figure 35 shows an example for an event and listener.

```
<?php
namespace App\Providers;

use Illuminate\Support\Facades\Event;
use Illuminate\Foundation\Support\Providers\EventServiceProvider as ServiceProvider;

class EventServiceProvider extends ServiceProvider
{
    /**
     * The event listener mappings for the application.
     *
     * @var array
     */
    protected $listen = [
        'App\Events\AnswerEvent' => [
            'App\Listeners\AnswerListener',
        ],
        'App\Events\QuestionEvent' => [
            'App\Listeners\QuestionListener',
        ],
        'App\Events\QuestionTypeEvent' => [
            'App\Listeners\QuestionTypeListener',
        ],
        'App\Events\UserEvent' => [
            'App\Listeners\UserListener',
        ],
        'App\Events\SurveyEvent' => [
            'App\Listeners\SurveyListener',
        ],
    ],
};
```

Figure 34 - Even Service Provide.

```

<?php
namespace App\Events;

use Illuminate\Broadcasting\Channel;
use Illuminate\Queue\SerializesModels;
use Illuminate\Broadcasting\PrivateChannel;
use Illuminate\Broadcasting\PresenceChannel;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;
use App\Answer;

class AnswerEvent implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;
    public $answer;

    /**
     * Create a new event instance.
     *
     * @return void
     */
    public function __construct(Answer $answer)
    {
        $this->answer = $answer;
    }

    /**
     * Get the channels the event should broadcast on.
     *
     * @return Channel|array
     */
    public function broadcastOn()
    {
        return new Channel('test-channel');
    }
}

```

```

<?php
namespace App\Listeners;

use App\Events\AnswerEvent;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Queue\ShouldQueue;
use Storage;

class AnswerListener
{
    /**
     * Create the event listener.
     *
     * @return void
     */
    public function __construct()
    {
        //
    }

    /**
     * Handle the event.
     *
     * @param AnswerEvent $event
     * @return void
     */
    public function handle(AnswerEvent $event)//se ejecuta
    {
        $message = $event->answer->answer.' Created. ';
        Storage::put('loginactivity.txt', $message);
    }
}

```

Figure 35 - Event and Listener.

4.2.7. Views

Views contain the HTML used by the application and separate the controller application logic from the presentation logic. Views were stored in the *resources/views* directory [57].

In the same way, views were made for each entity that was used in the Dashboard, to have the organized code, folders were created for each section that was available in the views. (see Figure 36).

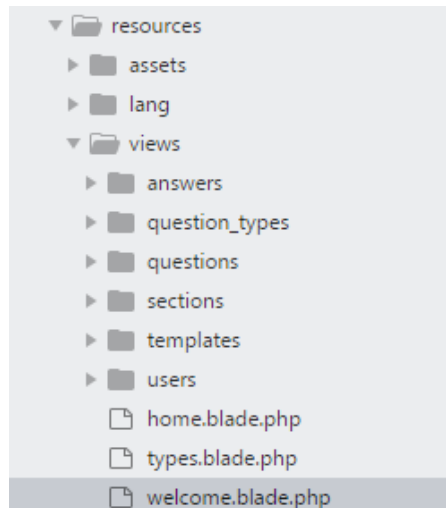


Figure 36 – Views in the project structure.

The main view contains a menu of options to display and insert data in the tables, it also contains information of certain tables and shows the graphs in real time (Figure 37).

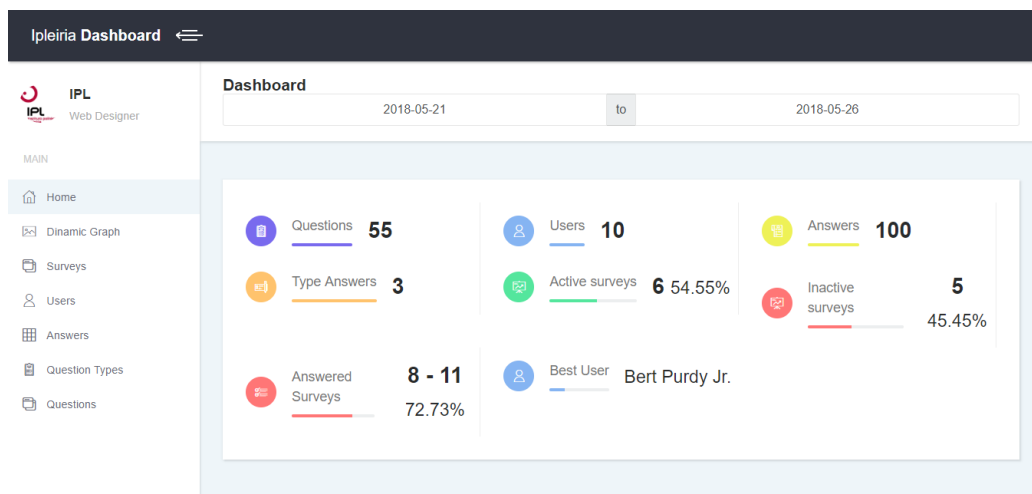


Figure 37 - Main Dashboard view.

Responsive web design

The Dashboard can be opened from any device since the design of the views was adaptable (see Figure 38).

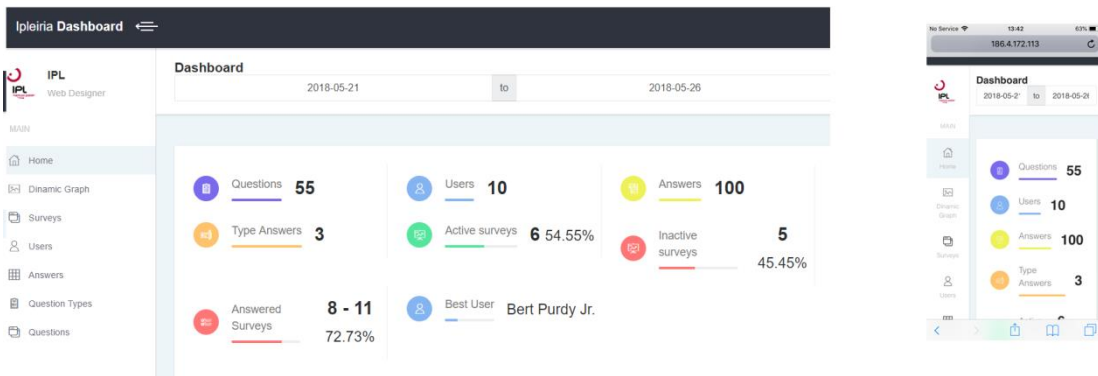


Figure 38 - Responsive Design.

4.2.8. Date picker to Temporal Analysis

In the main view A datepicker "bootstrap-datepicker" was implemented, start and end variables are configured to update the information and charts of the Dashboard

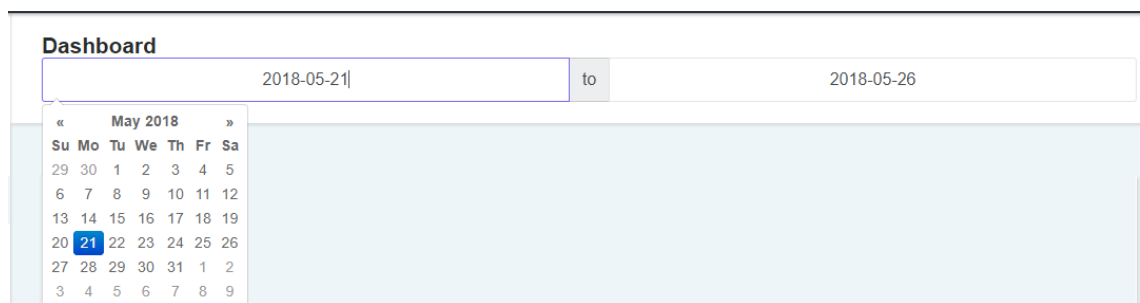


Figure 39 - Date picker for data range.

4.2.9. Date format

Better manipulation of the dates used in the Dashboard, the Carbon API was used, as follows (see Figure 40). This is used in the tables.

```

use Carbon\Carbon;

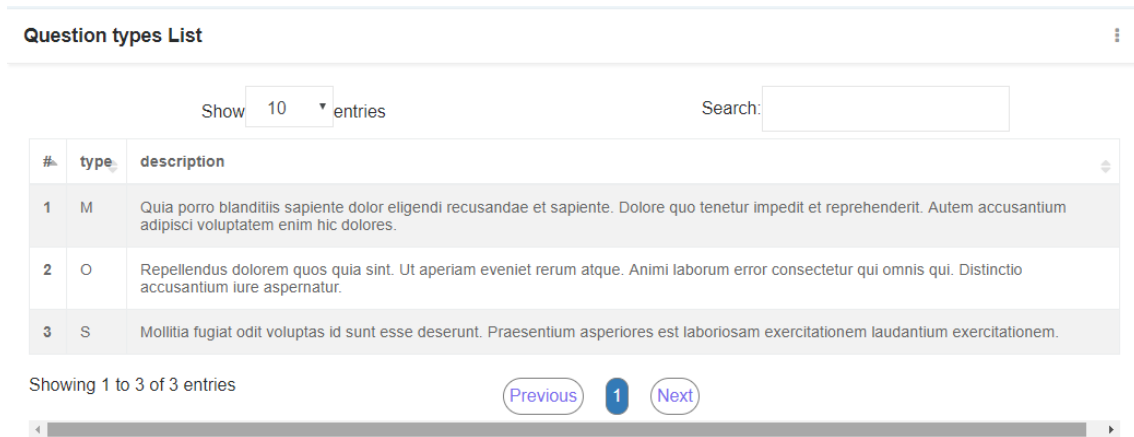
class Surveys extends Controller
{
    public function __construct()
    {
        Carbon::setLocale('pt');
    }
}

```

Figure 40 - Carbon Function.

4.2.10. Search, Sort and Pagination

To search, sorting and pagination a "jquery-Datatable [36]" was implemented in the list of data (see Figure 41).



The screenshot shows a web interface for a 'Question types List'. At the top, there is a search bar and a dropdown menu set to '10 entries'. Below this is a table with three columns: '#', 'type', and 'description'. The table contains three rows of data. At the bottom of the table, there is a pagination control showing 'Showing 1 to 3 of 3 entries' and buttons for 'Previous', '1', and 'Next'. A scrollbar is visible at the bottom of the table area.

#	type	description
1	M	Quia porro blanditiis sapiente dolor eligendi recusandae et sapiente. Dolore quo tenetur impedit et reprehenderit. Autem accusantium adipisci voluptatem enim hic dolores.
2	O	Repellendus dolorem quos quia sint. Ut aperiam eveniet rerum atque. Animi laborum error consectetur qui omnis qui. Distinctio accusantium iure aspernatur.
3	S	Mollitia fugiat odit voluptas id sunt esse deserunt. Praesentium asperiores est laboriosam exercitationem laudantium exercitationem.

Figure 41 - Search, Sort and Pagination.

4.2.11. Data Visualization used

In this research are used, 5 types of graphics which are detailed below:

- **Line chart** - Line Chart is drawn by interconnecting all data points in data series using straight line segments. Line Charts are normally used for visualizing trends in data varying continuously over a period of time or range [58].

Line chart is used to show the information about the number of questions answered by each user (see Figure 42).

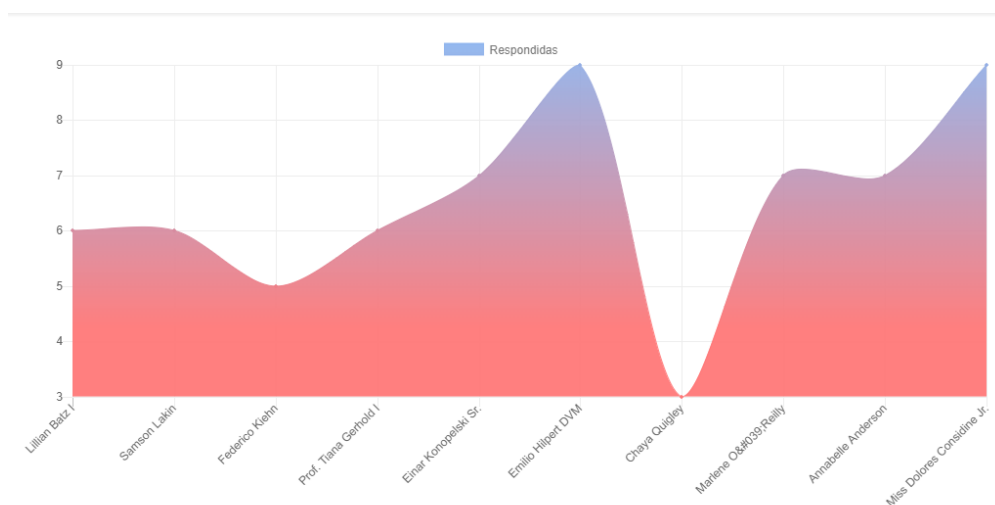


Figure 42 - Line chart of Questions by User.

In the configuration file, it was necessary to specify a function that calls the data used in the chart (see Figure 43).

```
function QuestionbyUser(){
$.get('{{route("questions_user')}}', {}, function(data){

var LINECHARTEXAMPLE = $('#lineChartExample');
var lineChartExample = new Chart(LINECHARTEXAMPLE, {
type: 'line',
options: {
legend: {labels:{fontColor:"#777", fontSize: 12}},
scales: {
xAxes: [{
display: true,
gridLines: {
color: '#eee'
}
}],
yAxes: [{
display: true,
gridLines: {
color: '#eee'
}
}
]}
},
data: {
labels: data.users,
datasets: [
{
```

Figure 43 - Function to graph the Line chart.

- **Pie chart** - Pie Chart divides a circle into multiple slices that are proportional to their contribution towards the total sum. Pie chart is useful in comparing the share or proportion of various items [59].

Pie chart was used to show the information about the existing amount of question types as type 'M', 'O', 'S' stored in the database (see Figure 44).

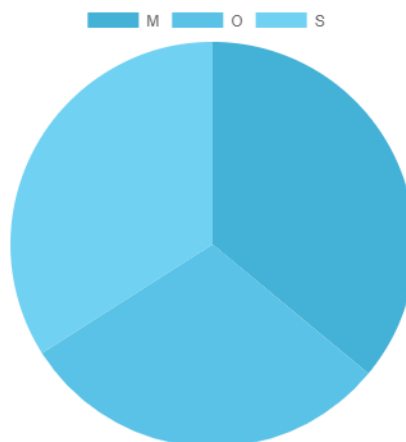


Figure 44 - Pie chart of Question Types.

The configuration file is in (see Figure 45).

```
function typesData(){
  $.get("{{route('tipos')}}", {}, function(data){
    var PIECHARTEXAMPLE = $('#pieChartExample');
    var pieChartExample = new Chart(PIECHARTEXAMPLE, {
      type: 'pie',
      data: {
        labels: data.tipos,
        datasets: [
          {
            data: data.questions,
            borderWidth: 0,
            backgroundColor: [
              '#44b2d7',
              '#59c2e6',
              '#71d1f2',
              '#96e5ff'
            ],
            hoverBackgroundColor: [
              '#44b2d7',
              '#59c2e6',
              '#71d1f2',
              '#96e5ff'
            ]
          }
        ]
      }
    });

    var pieChartExample = {
      responsive: true
    };

  }, 'json');
}
```

Figure 45 - Function to graph the Pie chart.

- **Radar chart** - In a radar chart, data points are drawn evenly spaced, clockwise around the chart. The value of the point is represented as the distance from the center of the chart, where the center represents the minimum value, and the chart edge is the maximum value. Each series is drawn as one complete circuit of the chart. The chart connects these points with straight or curved lines. So, a radar chart is essentially a line chart wrapped into a circle, where the y-axis goes from the center of the chart to the perimeter, and the x-axis is the perimeter of the chart, starting and ending at the 12:00 line[60].

Radar chart was used to show the average duration of the availability of a survey (see Figure 46).

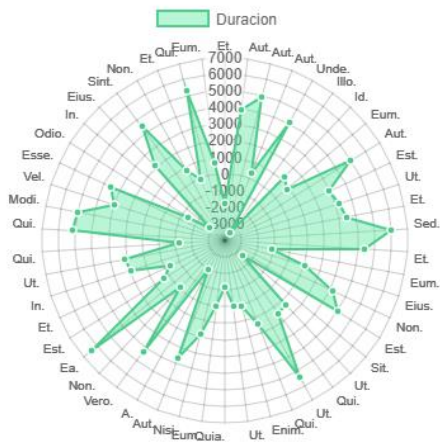


Figure 46 - Radar chart to show the Surveys Duration.

The configuration file is (see Figure 47).

```
function generateSurvey(){
    $.get('{{route("survey_list')}}', {}, function(data){

var RADARCHARTEXMPLE = $('#radarChartExample');
var radarChartExample = new Chart(RADARCHARTEXMPLE, {
    type: 'radar',
    data: {
        labels: data.surveys,
        datasets: [
            {
                label: "Duration",
                backgroundColor: "rgba(84, 230, 157, 0.4)",
                borderWidth: 2,
                borderColor: "rgba(75, 204, 140, 1)",
                pointBackgroundColor: "rgba(75, 204, 140, 1)",
                pointBorderColor: "#fff",
                pointHoverBackgroundColor: "#fff",
                pointHoverBorderColor: "rgba(75, 204, 140, 1)",
                data: data.surveys_time
            },
        ]
    }
});
var radarChartExample = {
    responsive: true
};
});
}
generateSurvey();
```

Figure 47 - Function to graph the Radar chart.

- **Bar chart** - Bar graph is represented by rectangular bars where length of bar is proportional to the values that they represent. It was used to compare values between different categories [61]. Bar chart was used to show the survey's count by users (see Figure 48).

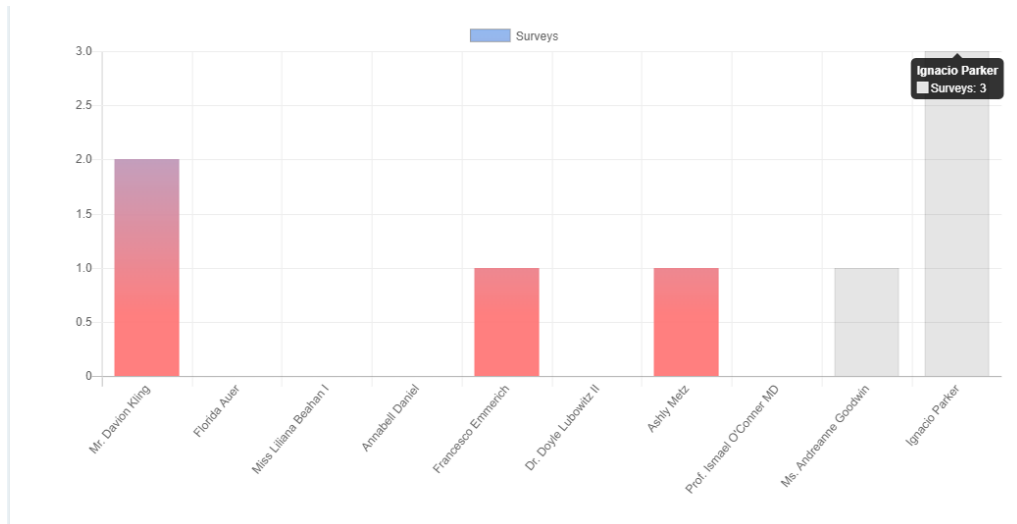


Figure 48 - Bar chart to show the Surveys Count.

The configuration file is (see Figure 49).

```
function generate_User_Survey(){
  $.get('{{route("user_surveys")}}', {}, function(data){

var BARCHARTEXAMPLE = $('#barChartExample');
var barChartExample = new Chart(BARCHARTEXAMPLE, {
  type: 'bar',
  options: {
    scales: {
      xAxes: [{
        display: true,
        gridLines: {
          color: '#eee'
        }
      }],
      yAxes: [{
        display: true,
        gridLines: {
          color: '#eee'
        }
      }],
    }, height : '500px'
  },
  data: {
    labels: data.users,
    datasets: [
      {
        label: "Surveys",
        backgroundColor: [
          gradient1,
          gradient1,
          gradient1,
          gradient1,
          gradient1,
          gradient1,
          gradient1
        ]
      }
    ]
  }
});
}
```

Figure 49 - Function to graph the Bar chart.

- **Doughnut Charts:** Are similar to pie charts except for a blank center. Doughnut Chart, also referred to as Donut Charts are useful to visually compare contribution of various items to the whole. Doughnut charts are beautiful, interactive, cross-

browser compatible, supports animation, exporting as image & real time updates [62].

The Doughnut Chart was used to show the anonymous surveys and non-anonymous (see Figure 50).

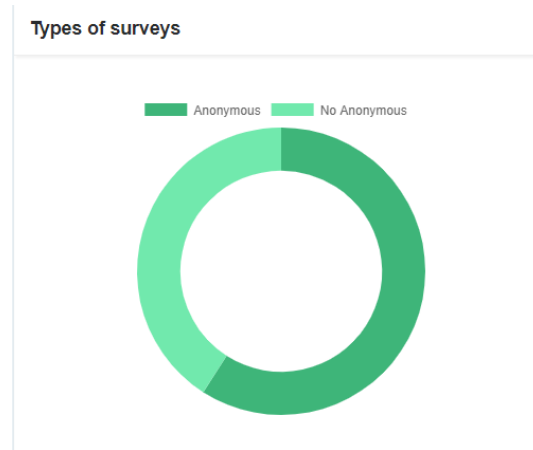


Figure 50 - Doughnut Chart to show the anonymous and non-anonymous surveys.

The configuration file is (see Figure 51).

```
function generate_Anonymous_Survey(){
  $.get('{{route("anonymous_surveys")}}', {}, function(data){

var DOUGHNUTCHARTEXAMPLE = $('#doughnutChartExample');
var pieChartExample = new Chart(DOUGHNUTCHARTEXAMPLE, {
  type: 'doughnut',
  options: {
    cutoutPercentage: 70,
  },
  data: {
    labels: [
      "Anonymous",
      "No Anonymous",
    ],
    datasets: [
      {
        data: [data.anonymous, data.no_anonymous],
        borderWidth: 0,
        backgroundColor: [
          '#3eb579',
          "#71e9ad"
        ],
        hoverBackgroundColor: [
          '#3eb579',
          "#71e9ad"
        ]
      }
    ]
  }
});
}, 'json');
```

Figure 51 - Function to graph the Doughnut Chart.

4.2.12. Real Time Communication

For the real time communication was necessary to use the next files and configurations:

- **Socket.js** - In this file the technologies used are defined as: express, socket.io, redis. It also specifies the port and channel used for communication (see Figure 52).

```

var app = require('express')();
var http = require('http').Server(app);
var io = require('socket.io')(http);
var Redis = require('ioredis');
//var redis = new Redis(3001, '192.168.1.42');
var redis = new Redis(3001);
redis.subscribe('test-channel', function(err, count) {
});

redis.on('message', function(channel, message) {
  console.log('Message Recieved: ' + message);
  message = JSON.parse(message);
  io.emit(channel + ':' + message.event, message.data);
});

//http.listen(3000, '192.168.1.42', function(){
  http.listen(3000, function(){
    console.log('Listening on Port 3001');
  });
});

```

Figure 52 - Socket.js content for communication.

- **Update the Dashboard view** - It is necessary to create a global variable that allows connection to the socket and redis-server, then the connection is made specifying the name of the channel, the associated event, the message, the function of the graph that must be updated or the part of the view that must be updated (see Figure 53).

```

var socket = io('http://127.0.0.1:3000');

socket.on("test-channel:App\\Events\\AnswerEvent", function(message){
  $('#totalanswer').text(parseInt($('#totalanswer').text()) + 1);
  QuestionbyUser();
});

```

Figure 53 - Function to connect.

4.3. Security Considerations

Some of these techniques were not implemented and can be considered as future work. The main ways to keep the Dashboard safe, are guaranteed with the technologies used in its development.

- **Laravel** - Laravel aims to make implementing authentication very simple. The authentication configuration file is located at *app/config/auth.php*, which contains several well documented options for tweaking the behavior of the authentication facilities. By default, Laravel includes a User model in your *app/models* directory which may be used with the default Eloquent authentication driver [26].
- **HTTP Basic Authentication** - HTTP Basic Authentication provides a quick way to authenticate users of the application without setting up a dedicated "login" page [26]. It was not implemented in the Dashboard.
- **Encryption** - Laravel's encrypter uses OpenSSL to provide AES-256 and AES-128 encryption. All of Laravel's encrypted values are signed using a message authentication code (MAC) so that their underlying value can not be modified once encrypted [26]. It was not implemented in the Dashboard.
- **Database** - By default, Laravel includes an *App\User* Eloquent model in the *app* directory. This model may be used with the default Eloquent authentication driver. If the application is not using Eloquent, use the database authentication driver which uses the Laravel query builder [26].
- **SQL injection** - Laravel's Eloquent ORM uses PDO parameter binding to avoid SQL injection. Parameter binding ensures that malicious users can't pass in query data which could modify the query's intent. Consider for instance a form field used to supply an e-mail address which might be used for searching a user table [63].
- **SSL Certificates** - When there is an own server for the dashboard it is possible to install the SSL certificates, to allow secure connections. Typically, SSL is used to

secure credit card transactions, data transfer and logins, and more recently is becoming the norm when securing browsing of social media sites [64].

This page was intentionally left blank

5. Production Environment, User Testing and Results

In this chapter, we present the production environment, detail the features of the server regarding the user testing, functionality testing and the results obtained in this research.

5.1. Project Web Hosting – Production Environment

Once the development was finished, the project was then uploaded to a server so that it is available to the user (see Appendix G).

The URL of the project is: <http://186.4.172.113/Dashboard-meicm/public/index.php>

The server was mounted on a home computer with the following features:

- Computer Pentium 4, 2.8 GHz
- Operative System Ubuntu 16.04
- 2GB ram
- 256 GB of Hard Disk
- Internet speed 3 MB
- Optical fiber provider Netlife
- Located in Quito Ecuador

5.2. Test and Results

Once the Dashboard was available to the users, it was necessary to perform several tests to ensure that it works correctly and is easy to use for the users. The following tests were carried out:

Functionality test - For the functionality test, the following criteria were considered:

- **Communication with the Database** - The communication is done through the `.env` file located in the root directory of the application. This file has the configuration data of the database;
- **Real-time update of the Dashboard** - This feature is met when using the redis server, socket, Laravel events and listeners;

- **Statistics** - In the home section, the most important statistics of a survey are shown;
- **Temporal Analysis** - It is possible to use date ranges to visualize the graphs and statistics;
- **Dynamic Graphics** - The user can select which data he wants to graph and which graphic to use;
- **Visualization of Text Surveys** - The content of the text type surveys is displayed;
- **Analysis of Numerical Surveys** - An analysis of the responses of a Numeric-type survey is carried out.

With all these features working properly, the success of the Dashboard is guaranteed in the analysis of survey results.

User tests - Once the development of the Dashboard is finished, it is hosted on a public server so that it can be used by users. The tests were conducted with 20 people where the following characteristics were taken into account:

- Is it understandable to the user?
- Is it easy to use?
- The techniques of Visualization used are adequate?
- How it is a free application, do you consider it is beneficial?
- Do you consider it important to conduct a survey analysis?

The results are summarized in the next table (see Table 2):

Questions	Yes	No	Total User
Is it understandable to the user?	17	3	20
Is it easy to use?	15	5	20
The techniques of Visualization used are adequate?	18	2	20

As it is a free application, do you consider it beneficial?	20	0	20
Do you consider it important to conduct a survey analysis?	20	0	20

Table 2 - Test Results.

The questions that are asked to the users are of general form what gives a global perspective of the use and operation of the Dashboard developed in this research.

The majority of users answered affirmatively to the questions "YES", so it can be said that the Dashboard developed is an excellent tool for the analysis of results and that being free and easy to use will be very well received by users.

The final result is a dynamic dashboard that uses several techniques of data visualization to show information of the surveys, it is generic, free and of easy use for the user.

This page was intentionally left blank

6. Conclusions and Future Work

In this research we conducted a state of the art review about Data visualization and identified a set of distinct Data Visualization techniques. A Dynamic Dashboard for Analyzing Surveys' Results was developed using the Laravel framework as the base for development.

Due to the massive amount of information that exists and is created every day, the number of existing artifacts and the needs of all users and consumers of knowledge out there, it is difficult to find ways to present the information in an accessible way or make sense out of it. Our proposal is to develop a Dashboard where users will be able to interact with the information, based on an initial set of hints, charts, tables and reports, produced by the Dashboard. This Dashboard will be free and easy to use by users.

Dashboards can provide a unique and powerful means to present information. No matter how great the technology, a Dashboard's success as a medium of communication, is a product of design, a result of a display that speaks clearly and immediately.

Regarding the implemented Dashboard, it is functional and can be used for the basic aim for which it was designed, but it can be improved with the implementation of: Authentication with different user profiles, features to allow exporting the data, adapting the model to other case studies – specially to scenarios where real time data is available, and, finally, more tests with end users, automated and load tests.

From this research work, a scientific article was published in the CISTI'2018 - 13th Iberian Conference on Information Systems and Technologies, that was held between the 13th and 16th of June 2018, at Cáceres, Spain. This article was presented at the Conference by Renato Toasa (see Appendix H).

This page was intentionally left blank

7. References

- [1] C. Min *et al.*, “Data, information, and knowledge in visualization,” *IEEE Comput. Graph. Appl.*, vol. 29, no. 1, pp. 12–19, 2009.
- [2] G. Bellinger, D. Castro, and A. Mills, “Data , Information , Knowledge , and Wisdom,” *Syst. Think.*, p. 5, 2004.
- [3] L. Chapman and E. Roberts, “[Point of contact].,” *Infirm. Can.*, vol. 20, no. 10, pp. 26–8, 1978.
- [4] Joseph DeVito, *Essentials of Human Communication: Joseph DeVito: 9780558323158: Amazon.com: Books*. 2009.
- [5] F. B. Viégas and M. Wattenberg, “Artistic Data Visualization: Beyond Visual Analytics,” *Online Communities Soc. Comput.*, vol. 4564, no. HCII 2007, LNCS 4564, pp. 182–191, 2007.
- [6] K. S. Khan, R. Kunz, J. Kleijnen, and G. Antes, “Five steps to conducting a systematic review,” *Jrsm*, vol. 96, no. 3, pp. 118–121, 2003.
- [7] M. Kintz, “A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology,” *CEUR Workshop Proc.*, vol. 947, pp. 31–36, 2012.
- [8] S. Few, “Clarifying the vision,” *Inf. Dashboard Des. Eff. Vis. Commun. Data*, p. 223, 2006.
- [9] R. Brath and M. Peters, “Dashboard Design: Why Design is Important,” *Data Min. Rev.*, 2004.
- [10] Smartsheet, “Identifying Which Dashboard Software Is Right,” 2017. [Online]. Available: <https://www.smartsheet.com/how-pick-best-dashboard-software-your-company>. [Accessed: 20-Jul-2009].
- [11] S. G. Archambault, “Charts & Infographics.”
- [12] “Zoho Reviews | TechnologyAdvice.” [Online]. Available: <http://technologyadvice.com/products/zoho-reviews/>. [Accessed: 09-Oct-2017].
- [13] “Sisense Reviews | TechnologyAdvice.” [Online]. Available: <http://technologyadvice.com/products/sisense-reviews/>. [Accessed: 09-Oct-2017].
- [14] “Business Dashboards: Data at a Glance.” [Online]. Available: <https://www.tableau.com/solutions/topic/business-dashboards>. [Accessed: 27-Sep-2017].

- [15] google, "Google Dashboard." [Online]. Available: <https://sites.google.com/a/pressatgoogle.com/googledashboard/>. [Accessed: 27-Sep-2017].
- [16] S. Rose, "Return on Information : The New ROI Getting value from data.," *SAS Inst. Inc. U.S.A*, 2014.
- [17] Operation-Meditation, "4 Powerful Visualization Techniques - Operation Meditation - Operation Meditation." [Online]. Available: <http://operationmeditation.com/discover/visualization-techniques/>. [Accessed: 10-Sep-2017].
- [18] Sas, "Data Visualization Techniques," *White Pap.*, pp. 2–16, 2013.
- [19] S. Yeh and K. Prussia, "NESUG 2007 And Now , Presenting ... Exploratory Visualization of Correlation Matrices," 2007.
- [20] "SAS Whitepaper Data Visualization Techniques - Documents." [Online]. Available: <https://docgo.org/sas-whitepaper-data-visualization-techniques>. [Accessed: 28-Sep-2017].
- [21] "Sankey Flow Show - Attractive flow diagrams made in minutes!" [Online]. Available: <http://www.sankeyflowshow.com/>. [Accessed: 28-Sep-2017].
- [22] "Data Visualization: Designing for Mobile Devices." [Online]. Available: <http://www.dummies.com/programming/big-data/big-data-visualization/data-visualization-designing-for-mobile-devices/>. [Accessed: 28-Sep-2017].
- [23] Canada Business Network, "Types of survey questions - Canada Business Network." [Online]. Available: <https://canadabusiness.ca/business-planning/market-research-and-statistics/conducting-market-research/types-of-survey-questions/>. [Accessed: 23-Mar-2018].
- [24] G. M. De Oliveira and M. G. Filipe, "DynVIX - Dynamic Visualization Infrastructure," 2017.
- [25] M. Alvarez *et al.*, "Application for monitoring primary energy resources based on open source software," in *Iberian Conference on Information Systems and Technologies, CISTI*, 2017.
- [26] "Laravel - The PHP Framework For Web Artisans." [Online]. Available: <https://laravel.com/docs/5.4/security>. [Accessed: 17-Apr-2018].
- [27] "MySQL :: MySQL 5.7 Reference Manual :: 1.3.1 What is MySQL?" [Online]. Available: <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>. [Accessed: 11-

Oct-2017].

- [28] “Introduction - Composer.” [Online]. Available: <https://getcomposer.org/doc/00-intro.md>. [Accessed: 04-Mar-2018].
- [29] “Artisan Console - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.6/artisan>. [Accessed: 07-Jun-2018].
- [30] “Package Management Basics: apt, yum, dnf, pkg | DigitalOcean.” [Online]. Available: <https://www.digitalocean.com/community/tutorials/package-management-basics-apt-yum-dnf-pkg>. [Accessed: 17-Jun-2018].
- [31] “The 15 Best JavaScript Charting Libraries.” [Online]. Available: <https://www.sitepoint.com/15-best-javascript-charting-libraries/>. [Accessed: 11-Oct-2017].
- [32] R. Van Dierendonck and S. Van Tienhoven, “D3 . JS : Data-Driven Documents.”
- [33] “Highcharts Javascript Charting Library - Highcharts.” [Online]. Available: <https://www.highcharts.com/products/highcharts/>. [Accessed: 12-Oct-2017].
- [34] “Bootstrap · The most popular HTML, CSS, and JS library in the world.” [Online]. Available: <https://getbootstrap.com/>. [Accessed: 07-Jun-2018].
- [35] “jQuery.” [Online]. Available: <https://jquery.com/>. [Accessed: 17-Jun-2018].
- [36] “DataTables | Table plug-in for jQuery.” [Online]. Available: <https://datatables.net/>. [Accessed: 17-Jun-2018].
- [37] “Redis.” [Online]. Available: <https://redis.io/documentation>. [Accessed: 20-Jan-2018].
- [38] “Node.js.” [Online]. Available: <https://nodejs.org/es/>. [Accessed: 20-Jan-2018].
- [39] “Socket.IO.” [Online]. Available: <https://socket.io/>. [Accessed: 20-Jan-2018].
- [40] “Font Awesome.” [Online]. Available: <https://fontawesome.com/>. [Accessed: 07-Jun-2018].
- [41] “HTML Responsive Web Design.” [Online]. Available: https://www.w3schools.com/html/html_responsive.asp. [Accessed: 14-Jun-2018].
- [42] “XAMPP Hosting.” [Online]. Available: <https://www.apachefriends.org/hosting.html>. [Accessed: 11-Oct-2017].
- [43] “PHP: ¿Qué es PHP? - Manual.” [Online]. Available: <http://php.net/manual/es/intro-what-is.php>. [Accessed: 04-Mar-2018].
- [44] “The Perl Programming Language - www.perl.org.” [Online]. Available:

- <https://www.perl.org/>. [Accessed: 13-Apr-2018].
- [45] “Drupal - Open Source CMS | Drupal.org.” [Online]. Available: <https://www.drupal.org/>. [Accessed: 13-Apr-2018].
- [46] “Joomla! The CMS Trusted By Millions for their Websites.” [Online]. Available: <https://www.joomla.org/>. [Accessed: 13-Apr-2018].
- [47] “Crea un sitio web asombroso en WordPress.com.” [Online]. Available: https://es.wordpress.com/create/?sgmt=gb&utm_source=adwords&utm_campaign=G_Search_Brand_Desktop_RoW_es_x_x&utm_medium=cpc&keyword=wordpress&creative=205478610420&campaignid=746830496&adgroupid=38619805586&matchtype=e&device=c&network=g&targetid=kwd-295456. [Accessed: 13-Apr-2018].
- [48] “Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/>. [Accessed: 13-Apr-2018].
- [49] “CodeIgniter Web Framework.” [Online]. Available: <https://codeigniter.com/>. [Accessed: 13-Apr-2018].
- [50] “Git.” [Online]. Available: <https://git-scm.com/>. [Accessed: 17-Jun-2018].
- [51] “Bitbucket | The Git solution for professional teams.” [Online]. Available: <https://bitbucket.org/product>. [Accessed: 17-Jun-2018].
- [52] “GitHub | Crunchbase.” [Online]. Available: <https://www.crunchbase.com/organization/github>. [Accessed: 17-Jun-2018].
- [53] “Database: Migrations - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.4/migrations>. [Accessed: 28-Dec-2017].
- [54] “Database: Seeding - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.4/seeding#writing-seeders>. [Accessed: 28-Dec-2017].
- [55] “Controllers - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.4/controllers>. [Accessed: 04-Jan-2018].
- [56] “Events - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.4/events>. [Accessed: 16-Apr-2018].
- [57] “Views - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.4/views>. [Accessed: 04-Jan-2018].
- [58] “JavaScript Line Charts & Graphs | CanvasJS.” [Online]. Available: <https://canvasjs.com/html5-javascript-line-chart/>. [Accessed: 07-Jan-2018].

- [59] “JavaScript Pie Charts & Graphs | CanvasJS.” [Online]. Available: <https://canvasjs.com/html5-javascript-pie-chart/>. [Accessed: 07-Jan-2018].
- [60] “Radar Charts | Image Charts | Google Developers.” [Online]. Available: https://developers.google.com/chart/image/docs/gallery/radar_charts. [Accessed: 07-Jan-2018].
- [61] “Bar Charts & Graphs | CanvasJS.” [Online]. Available: <https://canvasjs.com/html5-javascript-bar-chart/>. [Accessed: 02-Mar-2018].
- [62] “JavaScript Doughnut Charts & Graphs | CanvasJS.” [Online]. Available: <https://canvasjs.com/html5-javascript-doughnut-chart/>. [Accessed: 04-Mar-2018].
- [63] “Easy Laravel 5 Book - SQL Injection.” [Online]. Available: <https://www.easylaravelbook.com/blog/how-laravel-5-prevents-sql-injection-cross-site-request-forgery-and-cross-site-scripting/>. [Accessed: 17-Apr-2018].
- [64] C. Soghoian and S. Stamm, “Certified Lies: Detecting and Defeating Government Interception Attacks against SSL (Short Paper),” Springer, Berlin, Heidelberg, 2012, pp. 250–259.
- [65] “Database: Migrations - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.6/migrations>. [Accessed: 16-Jun-2018].
- [66] “Eloquent: Getting Started - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.6/eloquent>. [Accessed: 16-Jun-2018].
- [67] “Controllers - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.6/controllers>. [Accessed: 16-Jun-2018].
- [68] “Database: Seeding - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.6/seeding>. [Accessed: 16-Jun-2018].
- [69] “Events - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: <https://laravel.com/docs/5.6/events>. [Accessed: 16-Jun-2018].
- [70] “Laravel Artisan Route Command: The route:list Command - Stillat.” [Online]. Available: <https://stillat.com/blog/2016/12/07/laravel-artisan-route-command-the-routelist-command>. [Accessed: 17-Jun-2018].
- [71] “Package Control - the Sublime Text package manager.” [Online]. Available: <https://packagecontrol.io/>. [Accessed: 14-Jun-2018].
- [72] “Basic Git commands - Atlassian Documentation.” [Online]. Available: <https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html>. [Accessed: 17-Jun-2018].

- [73] “What is an API? (Application Programming Interface) | MuleSoft.” [Online]. Available: <https://www.mulesoft.com/resources/api/what-is-an-api>. [Accessed: 17-Jun-2018].
- [74] “What is business intelligence (BI)? - Definition from WhatIs.com.” [Online]. Available: <https://searchbusinessanalytics.techtarget.com/definition/business-intelligence-BI>. [Accessed: 17-Jun-2018].
- [75] “What is database (DB)? - Definition from WhatIs.com.” [Online]. Available: <https://searchsqlserver.techtarget.com/definition/database>. [Accessed: 17-Jun-2018].
- [76] “What is Data Visualization? - Definition from Techopedia.” [Online]. Available: <https://www.techopedia.com/definition/30180/data-visualization>. [Accessed: 17-Jun-2018].
- [77] “What is framework? - Definition from WhatIs.com.” [Online]. Available: <https://whatis.techtarget.com/definition/framework>. [Accessed: 17-Jun-2018].
- [78] “What is Linux operating system? - Definition from WhatIs.com.” [Online]. Available: <https://searchdatacenter.techtarget.com/definition/Linux-operating-system>. [Accessed: 17-Jun-2018].
- [79] “Reference Manager and Academic Social Network - Mendeley Database | Elsevier Solutions.” [Online]. Available: <https://www.elsevier.com/solutions/mendeley>. [Accessed: 18-Jun-2018].
- [80] “Real-time | Define Real-time at Dictionary.com.” [Online]. Available: <http://www.dictionary.com/browse/real-time>. [Accessed: 18-Jun-2018].
- [81] “Scientific Papers | Learn Science at Scitable.” [Online]. Available: <https://www.nature.com/scitable/topicpage/scientific-papers-13815490>. [Accessed: 17-Jun-2018].
- [82] “What Is A Survey (or Questionnaire)? | Qualtrics.” [Online]. Available: <https://www.qualtrics.com/experience-management/research/survey-basics/>. [Accessed: 17-Jun-2018].
- [83] “What is the Web? - Definition from Techopedia.” [Online]. Available: <https://www.techopedia.com/definition/5613/web>. [Accessed: 17-Jun-2018].
- [84] “What is a Web-Based Application? - Definition from Techopedia.” [Online]. Available: <https://www.techopedia.com/definition/26002/web-based-application>. [Accessed: 18-Jun-2018].
- [85] “What is Microsoft Windows? - Definition from WhatIs.com.” [Online]. Available:

<https://searchwindowserver.techtarget.com/definition/Windows>. [Accessed: 17-Jun-2018].

This page was intentionally left blank

Appendices

Appendix A

Reference Papers

The following table shows the most important investigations found and that serve as reference for the development of this research.

Title	Database	Citations	Autors
Information Exploration- Approach for Improving Operating Room Logistics and System Processes	Scholar Google	17	Paul G. Nagy, PhD, Ramon Konewko, MS, Max Warnock, Wendy Bernstein, MD, Jacob Seagull, PhD, Yan Xiao, PhD, Ivan George, BS, Adrian Park, MD
A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology`	Scholar Google	7	Maximilien Kintz
Information Dashboard design	Scholar Google	725	Stephen Few
What Do Technical Communicators Need to Know about Information Design?	Scholar Google	11	Karen Schriver
A Comparative Evaluation Between Two Design Solutions for an Information	Scholar Google	3	Lovisa Gannholm

Dashboard			
Dashboards, Charts & Infographics	Scholar Google	-	Susan Gardner Archambault
Digital Dashboard design using multiple data streams for disease surveillance with influenza surveillance as an example	Scholar Google	18	Calvin KY Cheng, MMedSc; Dennis KM Ip, MBBS, MPhil; Benjamin J Cowling, PhD; Lai Ming Ho, PhD; Gabriel M Leung, MD, MPH; Eric HY Lau, PhD
Dashboard Design: Why Design is Important	Scholar Google	21	Richard Brath and Michael Peters
The Visual Display of Quantitative Information	Scholar Google	10502	Edward Tufte
Dashboard Design for an Autonomous Car	Scholar Google	7	Nikhil Gowda, Kirstin Kohler, Wendy Ju
Data, information, and knowledge in visualization	Ieeexplore	208	Min Chen, David Ebert, Hans Hagen
Data Visualization : the State of the Art	Scholar Google	59	Frits h. Post, Gregory M. Nielson, Georges-Pierre bonneau
Artistic Data Visualization: Beyond Visual Analytics	Springer	131	Fernanda B. ViégasMartin Wattenberg
Visualizing high-dimensional data using t-	Scholar Google	2826	Laurens van der Maaten, Geoffrey

sne			Hinton
Survey of glyph-based visualization techniques for spatial multivariate medical data	Scopus-Elsevier	63	Timo Ropinski, Steffen Oeltze, BernhardPreim
Principal Manifolds for Data Visualization and Dimension Reduction	Springer	289	Alexander N. Gorban Balázs K6g1 Donald C. Wunsch Andrei Zinovyev
MDS-based Visual Survey of Biological Data Visualization Techniques	Scholar Google	-	A Kerren, K Kucher, F Scheiber
Massive Data Visualization Analysis Analysis of current visualization techniques and main challenges for the future	Ieeexplore	-	Manuel Pérez Cota, María Díaz Rodríguez, Miguel Ramón González-Castro
Improved data visualization techniques for analyzing macromolecule structural changes	Scholar Google	33	Jae Hyun Kim, Vidyashankara Iyer, Sangeeta B. Joshi, David B. Volkin, C. Russell Middaugh
Exploring incomplete data using visualization techniques	Springer	33	Matthias Templ, Andreas Alfons, Peter Filzmoser

Table 3 - Reference Papers.

Systematic Review

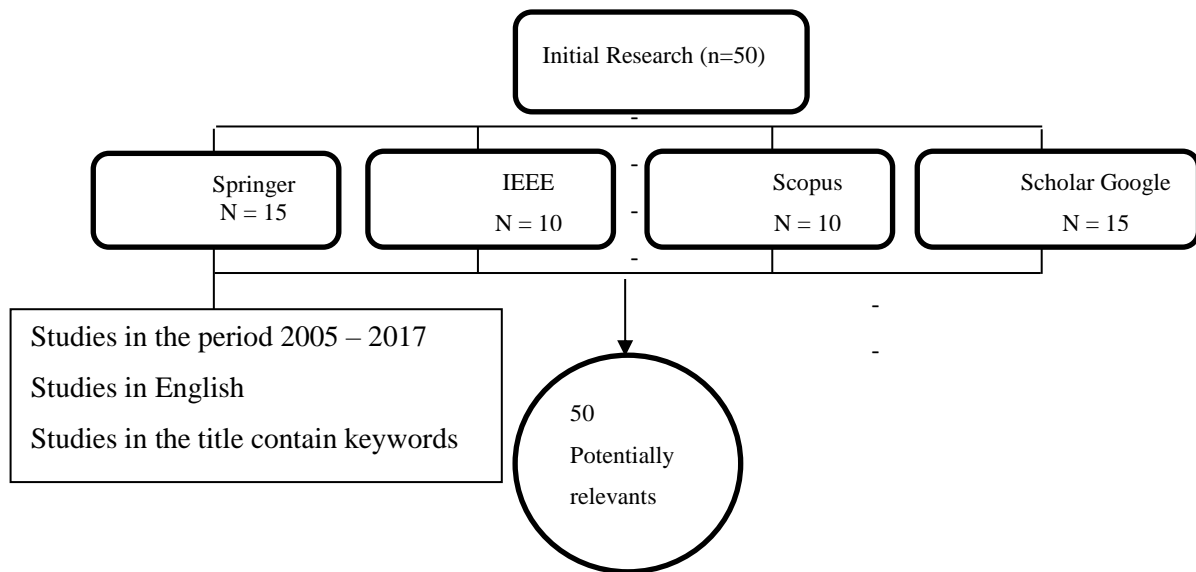


Figure 54 - Literature Review Results.

Appendix B

Artisan Comands

To create a new command, use the `make:command` Artisan command. This command will create a new command class in the `app/Console/Commands` directory.

- **make:migration** – Is to create a migration, the new migration will be placed in your `database/migrations` directory. Each migration file name contains a timestamp which allows Laravel to determine the order of the migrations [65].
- **make:model** - Is used to create an Eloquent model. Models typically live in the `app` directory, but you are free to place them anywhere that can be auto-loaded according to your `composer.json` file. All Eloquent models extend `Illuminate\Database\Eloquent\Model` class [66].
- **make:controller** - This command will generate a controller at `app/Http/Controllers/PhotoController.php`. The controller will contain a method for each of the available resource operations [67].
- **make:seeder** - Is to generate a seeder, execute the `make:seeder` Artisan command. All seeders generated by the framework will be placed in the `database/seeds` directory [68].
- **make:event** - This command will generate any events or listeners that are listed in your `EventServiceProvider`. Of course, events and listeners that already exist will be left untouched [69].
- **migrate:refresh** - Rollback all migrations and run them all again [53].
- **route:list** - The `route:list` command can be used to show a list of all the registered routes for the application. This command will display the domain, method, URI, name, action and middleware for the routes it includes in the generated table [70].

Appendix C

Sublime Text Packages

Sublime text offers the developer the option to install packages to improve the software development environment, the packages used in this research are [71]:

- **Package Control** - It is basically the main of all the plugins, from it we can manage the installation, uninstallation, editing and much more than other plugins, quickly and, above all, very intuitive;
- **CodeFormattter** - CodeFormatter is a Sublime Text 2/3 plugin that supports format (beautify) source code. CodeFormatter has support for the following languages: PHP, JavaScriptHTML, CSS, LESS, SASS, Python Visual Basic/VBScript, Coldfusion/Railo/Lucee;
- **Auto-completed** - Extend Sublime autocompletion to find matches in all open files of the current window;
- **PHPDocumentor** - Is a Sublime Text 2/3 plugin to speedup writing documenting comments;
- **JavaScript Completions** - JavaScript Completions for sublime text. It helps you to write your scripts more quickly with hints and completions;
- **SideBarEnhancements** - Enhancements to Sublime Text sidebar. Files and folders.

Appendix D

Git commands

The commands used to control versions of this research are [72]:

- **git init** - Create a new local repository;
- **git clone** - Create a working copy of a local repository;
- **git add** - Add one or more files to staging;
- **git commit** - Commit changes to head;
- **git push** - Send changes to the master branch of your remote repository;
- **git status** - List the files you've changed and those you still need to add or commit;
- **git pull** - Fetch and merge changes on the remote server to your working directory;
- **git merge** - To merge a different branch into the active branch.

Appendix E

Prototype

Before starting the development of the Dashboard, a basic prototype was made of how the Dashboard will look in its final version (see the next Figures), for this the balsamiq software was used.

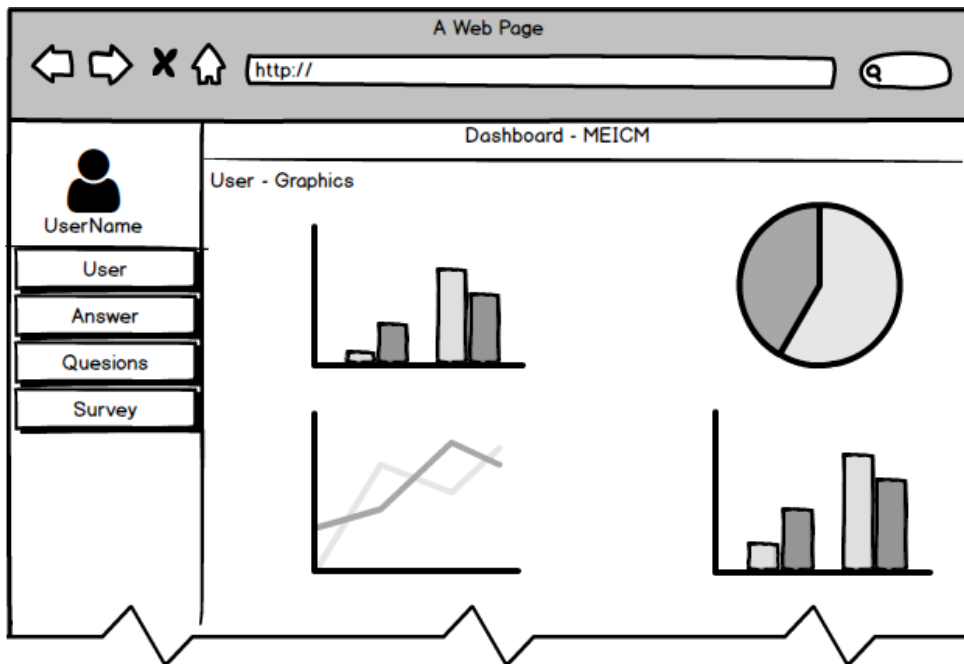


Figure 55 - Prototype - User View.

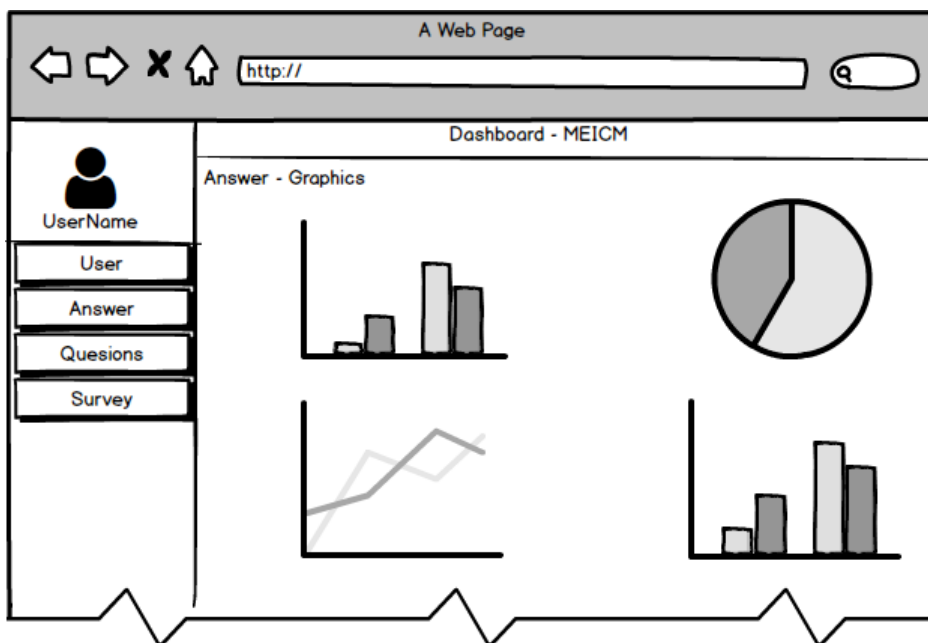


Figure 56 - Prototype - Answer View.

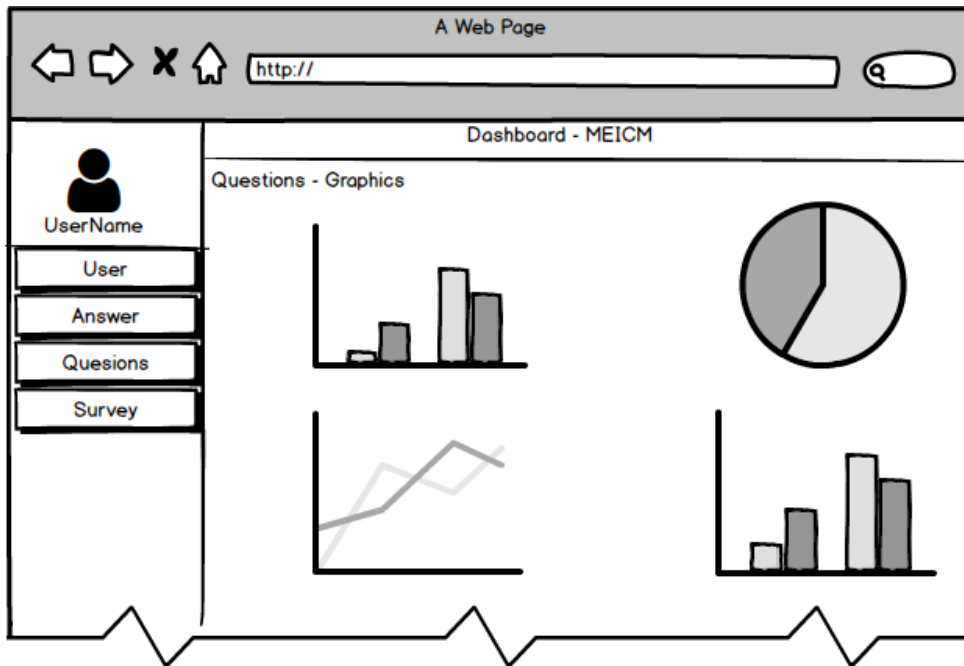


Figure 57 - Prototype - Questions View.

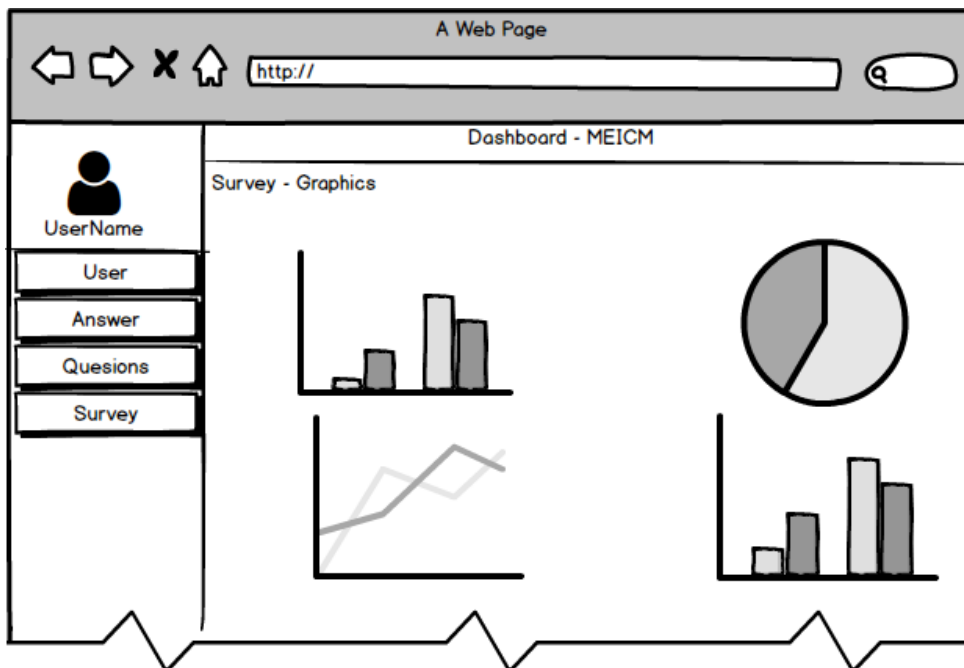


Figure 58 - Prototype - Surveys View.

Appendix F

Work environment preparation

On Windows:

First download and install XAMPP (see Figure 59).

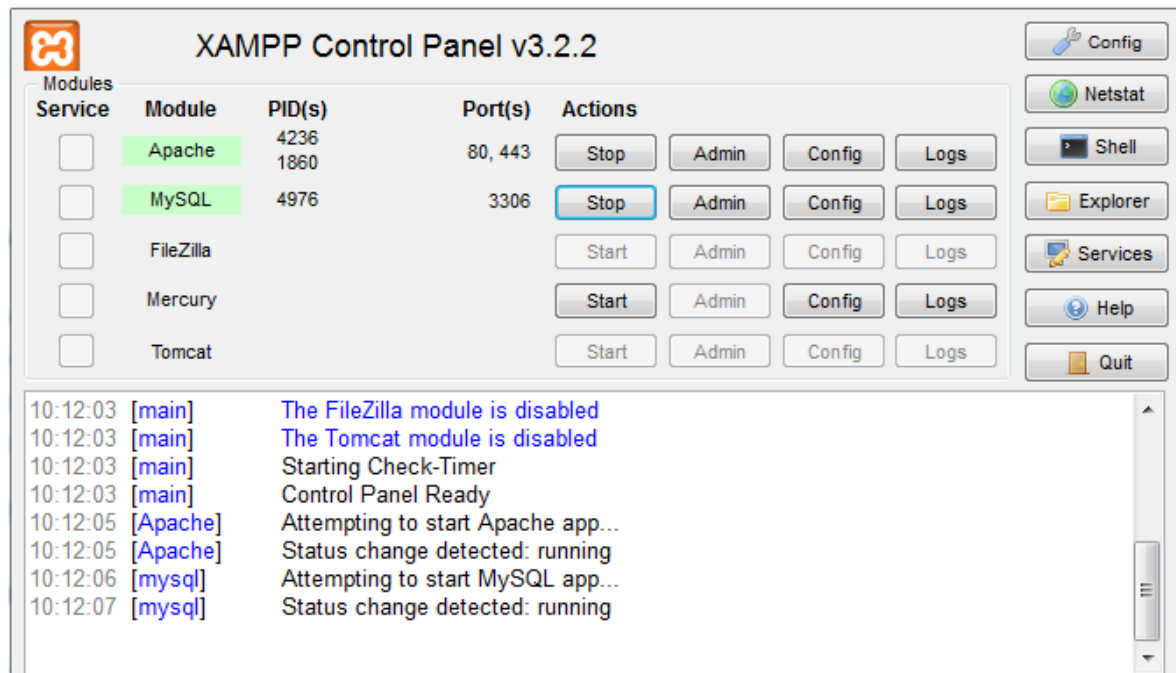


Figure 59 - Xampp Control Panel.

Install Composer

Composer is the dependency manager of PHP. From which we can easily download Laravel and all its dependencies

On Linux:

- Open a console and run the following commands for install Apache, PHP and MySQL:

```
sudo apt-get install apache2
```

```
sudo apt-get install php5 libapache2-mod-php5
```

```
sudo /etc/init.d/apache2 restart
```

```
sudo apt-get install mysql-server
```

```
sudo apt-get install php5-mysql php5-curl php5-gd php5-idn php-pear php5-imagick php5-imap  
php5-mcrypt php5-memcache php5-ming php5-ps php5-pspell php5-recode php5-snmp php5-  
sqlite php5-tidy php5-xmlrpc php5-xsl
```

```
sudo /etc/init.d/apache2 restart
```

```
sudo apt-get install phpmyadmin
```


- Install Composer with the following commands

```
curl -sS https://getcomposer.org/installer | php  
mv composer.phar /usr/local/bin/composer
```

Create a Laravel Project

Once we have all of the above installed, we will have to download Laravel from Composer, typing in the console the following commands

```
cd C:/xampp/htdocs  
composer create-project laravel/laravel Dashboard-meicm
```

This instruction will take a while as you have to download Laravel and all its dependencies (see Figure 60). When it finishes, you have Laravel installed on the server to start programming.

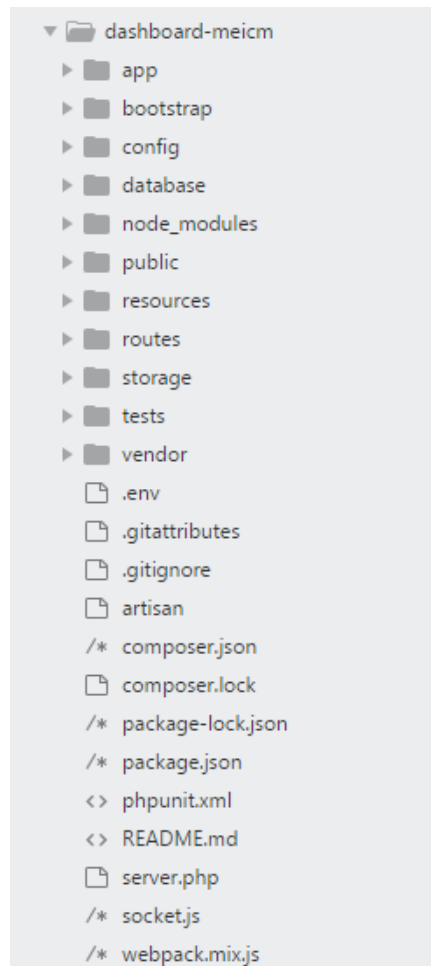


Figure 60 - Laravel dependencies.

Database Configuration

Laravel interacts with databases in an extremely simple way and across a variety of database backends using either raw SQL, the fluent query builder, and the Eloquent ORM. Currently, Laravel supports four databases[54]:

MySQL

PostgreSQL

SQLite

SQL Server

In this research, we will work with MySQL.

The database configuration for the application is located at config/database.php. In this file can define all of the database connections, as well as specify which connection should be used by default (see Figure 61 and Figure 62).

```
'default' => env('DB_CONNECTION', 'mysql'),

/*
-----
Database Connections
-----
Here are each of the database connections setup for your application.
Of course, examples of configuring each database platform that is
supported by Laravel is shown below to make development simple.

All database work in Laravel is done through the PHP PDO facilities
so make sure you have the driver for your particular database of
choice installed on your machine before you begin development.
*/

'connections' => [

    'sqlite' => [
        'driver' => 'sqlite',
        'database' => env('DB_DATABASE', database_path('database.sqlite')),
        'prefix' => '',
    ],

    'mysql' => [
        'driver' => 'mysql',
        'host' => env('DB_HOST', '127.0.0.1'),
        'port' => env('DB_PORT', '3306'),
        'database' => env('DB_DATABASE', 'forge'),
        'username' => env('DB_USERNAME', 'forge'),
        'password' => env('DB_PASSWORD', ''),
        'unix_socket' => env('DB_SOCKET', ''),
        'charset' => 'utf8mb4',
        'collation' => 'utf8mb4_unicode_ci',
        'prefix' => '',
        'strict' => true,
        'engine' => null,
    ],
],
```

Figure 61 - Database.php file.

Another important file is .env and is located in the root of project, in this file also can define the database connection (see Figure 62).

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:jbC4Lys7JdRis+FhSLB9rQXM+CxrDM5MhdqzkT1q2I=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dashboard
DB_USERNAME=root
DB_PASSWORD=12345
```

Figure 62 - .env file.

Appendix G

Running the Project on a Server

- Install Apache, PHP, MySQL
- Clone or Copy the project in:
 - Linux /var/www/html/*
 - Windows Xampp/htdocs*
- Give write permissions to the storage folder, the storage directory and the bootstrap/cache folder must have write permissions by the web server. :
 - sudo chmod -R 777 storage - bootstrap*
- In the project execute: *composer install*
- Install Nodejs
- Install Redis-server
- Create .env File
- In the project execute *php artisan migrate:refresh --seed*
- Depending on the IP addresses of the new server, change Ip Directions in the files:
 - Project/resources/views/templates/base.blade.php*
 - Project/socket.js*
- Execute Socket.js
- Execute redis-server
- In a browser open <http://IP-SERVER/Dashboard-meicm/public/index.php>
- OK.

Appendix H

Paper published at an international conference CISTI 2018.

Data Visualization Techniques for real-time information - A Custom and Dynamic Dashboard for Analyzing Surveys' Results

Renato Toasa^{1,2}, Marisa Maximiano¹, Catarina Reis¹, David Guevara²

¹Computer Science and Communication Research Centre (CIIC), Polytechnic Institute of Leiria, Portugal
2162342@my.ipleiria.pt, {marisa.maximiano, catarina.reis}@ipleiria.pt

²Facultad de Ingenieria en Sistemas, Electronica e Industrial, Universidad Tecnica de Ambato, Ecuador
dguevara@uta.edu.ec

Abstract — To achieve the most understandable and accurate display of information, a study on the available techniques of data visualization for real-time information must be made. Customizing existing platforms and designing specific boards, are among the important task to perform an accurate visualization of the information. In this paper, we conduct a literature review of data visualization, its techniques and existing Dashboard platforms. We implemented a generic and dynamic Dashboard based on real-time information with the aim to assess the impact of the available Data Visualization Techniques in the developed Dashboard. Therefore, our Dashboard users will be able to interact with the information, based on an initial set of hints, charts, tables and reports, produced by the Dashboard itself. This will allow us to test an existing set of data visualization techniques and create a new tailored Dashboard, showing that Dashboards can become a unique and powerful means to provide information.

Keywords - Data visualization techniques; Dashboard; real-time information; platform; tailored

This page was intentionally left blank

Glossary

Application Programming Interface (API): Is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API [73].

Business Intelligence (BI): Is a technology-driven process for analyzing data and presenting actionable information to help executives, managers and other corporate end users make informed business decisions [74].

Database: A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it [75].

Data Visualization: Data visualization is the process of displaying data/information in graphical charts, figures and bars. It is used as means to deliver visual reporting to users for the performance, operations or general statistics of an application, network, hardware or virtually any IT asset [76].

Framework: In computer systems, a framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate. Some computer system frameworks also include actual programs, specify programming interfaces, or offer programming tools for using the frameworks. A framework may be for a set of functions within a system and how they interrelate; the layers of an operating system; the layers of an application subsystem; how communication should be standardized at some level of a network; and so forth. A framework is generally more comprehensive than a protocol and more prescriptive than a structure [77].

Linux: Is a Unix-like, open source and community-developed operating system for computers, servers, mainframes, mobile devices and embedded devices. It is supported on almost every major computer platform including x86, ARM and SPARC, making it one of the most widely supported operating systems [78].

Mendeley: Is a free reference manager and academic social network that can help to organize a research, collaborate with others online, and discover the latest research, automatically generate bibliographies, collaborate easily with other researchers online, easily import papers from other research software, find relevant papers based on what you're reading, read papers on the go, with iOS and Android apps [79].

Real Time: Is the feature of the applications in which the computer must respond as rapidly as required by the user or necessitated by the process being controlled [80].

Scientific paper: Scientific papers are documents for sharing your own original research work with other scientists or for reviewing the research conducted by others. As such, they are critical to the evolution of modern science, in which the work of one scientist builds upon that of others. To reach their goal, papers must aim to inform, not impress. They must be highly readable — that is, clear, accurate, and concise. They are more likely to be cited by other scientists if they are helpful rather than cryptic or self-centered [81].

Survey: A survey is a method of gathering information from a sample of people, traditionally with the intention of generalizing the results to a larger population. Surveys provide a critical source of data and insights for nearly everyone engaged in the information economy, from businesses and the media to government and academics [82].

Web: The Web is the common name for the World Wide Web, a subset of the Internet consisting of the pages that can be accessed by a Web browser. Many people assume that the Web is the same as the Internet, and use these terms interchangeably. However, the term Internet actually refers to the global network of servers that makes the information sharing that happens over the Web possible. So, although the Web does make up a large portion of the Internet, but they are not one and same [83].

Web-based: A web-based application is any program that is accessed over a network connection using HTTP, rather than existing within a device's memory. Web-based applications often run inside a web browser. However, web-based applications also may be client-based, where a small part of the program is downloaded to a user's desktop, but processing is done over the internet on an external server. Web-based applications are also known as web apps [84].

Windows O/S: Is Microsoft's flagship operating system, the de facto standard for home and business computers. The GUI-based OS was introduced in 1985 and has been released in many versions since then, as described below. Microsoft got its start with the partnership of Bill Gates and Paul Allen in 1975. Gates and Allen co-developed Xenix (a version of Unix) and also collaborated on a BASIC interpreter for the Altair 8800. The company was incorporated in 1981 [85].