

UNIVERSITY OF THESSALY

Visualizing Next Generation Power Grid

Author:

Lamprini Vasilaki

Supervisor:

Manolis Vavalis,

Professor University of Thessaly

*A Thesis submitted for the degree of
Diploma of Science in
Computer and Communication Engineering*



University of Thessaly
Department of Electrical and Computer Engineering
Volos, Greece

October 2015

VISUALIZING NEXT GENERATION POWER GRID

ΟΠΤΙΚΟΠΟΙΩΝΤΑΣ ΔΙΚΤΥΑ ΕΝΕΡΓΕΙΑΣ ΝΕΑΣ ΓΕΝΙΑΣ

By

Lamprini Vasilaki

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

DIPLOMA OF SCIENCE

In Computer and Communication Engineering

UNIVERSITY OF THESSALY

2015

© 2015 Lamprini Vasilaki



Declaration of Authorship

I, Lamprini Vasilaki hereby certify that this thesis titled, “Visualizing Next Generation Power Grid” and the work presented in it has been composed by me and is based on my own work, unless stated otherwise. The research was carried out wholly or mainly while in candidature for the graduate degree of Diploma of Science in Computer and Communication Engineering at the University of Thessaly, Department of Electrical and Computer Engineering, Greece. Wherever I have consulted or quoted from the work of others, it is always attributed and the source is given. The main sources of help are referenced in the Bibliography section of this thesis.

Copyright © 2015 by Lamprini Vasilaki

“The copyright of this thesis rests with the author. No quotations from it should be published without the authors’ prior written consent and information derived from it should be acknowledged”.



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ &
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΟΠΤΙΚΟΠΟΙΩΝΤΑΣ ΔΙΚΤΥΑ ΕΝΕΡΓΕΙΑΣ
ΝΕΑΣ ΓΕΝΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΛΑΜΠΡΙΝΗΣ ΒΑΣΙΛΑΚΗ

Επιβλέπων : Μανόλης Βάβαλης
Καθηγητής Τ.Η.Μ.Μ.Υ.

Εγκρίθηκε από τη διμελή εξεταστική επιτροπή τον Οκτώβριο 2015.

(Υπογραφή)

.....
Μανόλης Βάβαλης
Καθηγητής Τ.Η.Μ.Μ.Υ.

(Υπογραφή)

.....
Ηλίας Χούστης
Ομότιμος Καθηγητής Τ.Η.Μ.Μ.Υ.

Βόλος, Οκτώβριος 2015

(Υπογραφή)

.....

**ΛΑΜΠΡΙΝΗ
ΒΑΣΙΛΑΚΗ**

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων
Πανεπιστημίου Θεσσαλίας

Copyright © Lamprini Vasilaki, 2015

Με επιφύλαξη παντός δικαιώματος, All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς το συγγραφέα

Περίληψη

Στόχος της διπλωματικής αυτής εργασίας είναι ο σχεδιασμός, η υλοποίηση και η κατ' αρχήν αξιολόγηση μιας εφαρμογής με βάση το διαδίκτυο, η οποία λαμβάνει δεδομένα από ένα σύστημα προσομοίωσης διανομής ηλεκτρικής ενέργειας (GridLAB-D) και τα αναπαριστά σε μορφή τέτοια ώστε να είναι κατανοητά από το χρήστη.

Τα δεδομένα που λαμβάνονται από το πρόγραμμα προσομοίωσης είναι υψηλής σημασίας και πολύ μεγάλης ακρίβειας. Ταυτόχρονα όμως είναι πολύ μεγάλος ο όγκος των δεδομένων, πράγμα που αποθαρρύνει έναν απλό χρήστη από το να τα ελέγξει ώστε να αντλήσει τις πληροφορίες που τον ενδιαφέρουν.

Σκοπός της εν λόγω εφαρμογής είναι να παρουσιάσει τα ανεπεξέργαστα και υψηλής αξίας δεδομένα, με έξυπνο και αποτελεσματικό τρόπο χρησιμοποιώντας τεχνικές οπτικοποίησης δεδομένων, έτσι ώστε ο χρήστης να είναι σε θέση να βλέπει ακριβώς τα δεδομένα που επιθυμεί ανά πάσα χρονική στιγμή, χωρίς να έχει αμφιβολίες για την ορθότητα ή την ακρίβειά τους. Η μορφή των δεδομένων θα επιλέγεται κάθε φορά από τον χρήστη, από ένα σύνολο υποστηριζόμενων μορφών.

Μια ακόμη πρόκληση της εφαρμογής είναι ο χρήστης να μπορεί να περιηγηθεί σε αυτή χωρίς τη βοήθεια κάποιου οδηγού, απλά διαβάζοντας τις περιγραφές των σελίδων της και ακολουθώντας τη ροή της.

Abstract

The main objective of this diploma thesis project is to design, implement and perform a preliminary evaluation of a web based application, which takes valuable data from a power distribution simulation system (GridLAB-D) and represents them in a form such that it is easily understood by the user.

The data obtained from the simulation program are of high importance and very high precision. Nevertheless they are large, and this discourages a naïve or even experienced user from examining them and understanding the information of interest to him.

The purpose of this application is to present the raw and valuable data in an intelligent and efficient manner using modern data visualization techniques, so that the user is able to just view the data they want at any desired time, without any doubt on the correctness or accuracy. The data format will be selected each time by the user from a predefined set of supported data formats.

Another challenge of this application is that the user may navigate to it without the help of a wizard, just reading the descriptions of its pages and following the flow of application.

Table of Contents

1. Introduction.....	20
1.1. Introduction to Big Data & Data Visualization Techniques.....	20
1.2. About this diploma thesis project.....	21
1.3. Content's Organization.....	21
2. Data Visualization.....	22
2.1. What is Big Data?.....	22
2.2. What is Data Visualization?.....	24
2.3. Visualization Techniques.....	25
2.3.1. 1 - Dimension or Linear.....	25
2.3.2. 2 - Dimension or Planar.....	25
2.3.2.1. Choropleth Map.....	25
2.3.2.2. Dasymetric Map.....	26
2.3.2.3. Cartogram.....	27
2.3.2.4. Dot Distribution Map.....	27
2.3.2.5. Proportional Symbol.....	28
2.3.2.6. Contour/ Isopleth/ Isarithmic Map.....	29
2.3.3. 3-Dimension or Volumetric.....	30
2.3.3.1. 3D Modeling.....	30
2.3.3.2. Computer Simulation.....	30
2.3.4. Temporal.....	31
2.3.4.1. Timeline.....	31
2.3.4.2. Time Series.....	32
2.3.4.3. Gantt Chart.....	32
2.3.4.4. Streamgraph.....	33
2.3.4.5. Arc Diagram.....	33

2.3.4.6. Sankey Diagram.....	34
2.3.4.7. Alluvial Diagram.....	34
2.3.5. Multidimensional.....	35
2.3.5.1. Tag Cloud.....	35
2.3.5.2. Tree Map.....	35
2.3.5.3. Scatter Plot.....	36
2.3.5.4. Bubble Chart.....	37
2.3.5.5. Heatmap.....	37
2.3.5.6. Parallel Coordinates.....	38
2.3.5.7. Box Plot.....	38
2.3.5.8. Line Plot.....	39
2.3.5.9. Radar/ Spider Chart.....	39
2.3.6. Tree/ Hierarchical.....	40
2.3.6.1. Dendrogram.....	40
2.3.6.2. Radial Tree.....	40
2.3.6.3. Hyperbolic Tree.....	41
2.3.7. Networks.....	42
2.3.7.1. Circular Hierarchy/ Dependency Graph.....	42
2.3.7.2. Node – link Diagram.....	42
2.4. Data Visualization Tools.....	43
2.4.1. SVG & HTML 5 Canvas.....	43
2.4.1.1. SVG.....	43
2.4.1.2. HTML 5 Canvas.....	43
2.4.2. JavaScript Libraries.....	44
2.4.2.1. General Libraries.....	44
2.4.2.1.1. D3.js.....	44

2.4.2.1.2. Highcharts.....	44
2.4.2.1.3. Fusion Charts.....	44
2.4.2.1.4. Charts.js.....	45
2.4.2.1.5. Google Charts.....	45
2.4.2.1.6. Raphaël.....	45
2.4.2.2. Graph Visualization Libraries.....	45
2.4.2.2.1. Vis.js.....	46
2.4.2.2.2. Graph Dracula.....	46
2.4.2.2.3. jsPlumb.....	46
2.4.2.2.4. Sigma.js.....	46
2.4.2.2.5. Cytoscape.js.....	46
2.4.2.3. JavaScript Libraries for Maps.....	47
2.4.2.3.1. Leaflet.....	47
2.4.2.3.2. Polymaps.....	47
2.4.2.3.3. Open Layers.....	47
2.4.2.3.4. Kartograph.....	47
2.4.2.3.5. jHERE.....	47
2.4.2.4. Time Series Visualization Libraries.....	48
2.4.2.4.1. MetricsGraphics.js.....	48
2.4.2.4.2. Cubism.js.....	48
2.4.2.4.3. Crossfilter.....	48
2.4.2.4.4. Rickshaw.....	48
3. System Design.....	49
3.1. Organizing of our data.....	49
3.2. Data Processing & Database formation.....	53
4. System Implementation.....	55

4.1 Architecture.....	55
4.2. Used Technologies.....	56
4.2.1. Backend Technologies.....	56
4.2.1.1. Node.js.....	56
4.2.1.2. Express.js.....	56
4.2.1.3. Sails.js.....	57
4.2.1.4. MySQL.....	57
4.2.1.5. Npm.....	57
4.2.2. Frontend Technologies.....	57
4.2.2.1. HTML/ HTML 5.....	57
4.2.2.2. CSS/ CSS 3.....	58
4.2.2.3. JavaScript.....	58
4.2.2.4. AngularJS.....	58
4.2.2.5. Twitter Bootstrap.....	58
4.2.2.6. jQuery.....	59
4.2.2.7. AJAX.....	59
4.2.2.8. JSON.....	59
4.3. MVC Architecture.....	60
4.4. User Interface.....	61
5. Differences with existing tools.....	85
6. Synopsis & Future Work.....	87
6.1. Synopsis.....	87
6.2. Future Work.....	87

Table of Figures

Figure 1 - An infographic about the four v's of Big Data (source: https://tomyrhymond.files.wordpress.com/2014/08/big-data.png).....	23
Figure 2 - Choropleth map that visualizes the population per square mile by state (source: http://katieschoettler.blogspot.gr/2010/04/standardized-choropleth-maps.html).....	26
Figure 3 - Dasymetric map of climate and plant hardiness zones (source: https://upload.wikimedia.org/wikipedia/commons/b/bf/USDA_Hardiness_zone_map.jpg).....	26
Figure 4 - A cartogram of 2012 electoral votes (source: http://stko.geog.ucsb.edu/node/11).....	27
Figure 5 - A one-to-one dot map of U.S. Public Libraries (source: http://gothos.info/wp-content/uploads/2013/03/publibs_2009.png).....	28
Figure 6 - A one-to-many dot map of Walmarts in the United States in 2009 (source: http://enb105-2012s-kst.blogspot.gr/2012/02/types-of-maps.html).....	28
Figure 7 - A proportional symbol map of wine consumption in Western Europe in 2010 (source: http://gis8jake.blogspot.gr/2013/03/cartography-lab-8-proportional-symbol.html).....	29
Figure 8 - A contour map of 500 millibar height in 1982 (source: https://upload.wikimedia.org/wikipedia/commons/5/5b/January_17_1982_500-Millibar_Height_Contours.png).....	29
Figure 9 - A 3D computer graphics model (source: http://www.cgarena.com/freestuff/tutorials/max/ridley/04-Modeling_RidleyHead_wip.jpg).....	30
Figure 10 - Computer simulation of the process of osmosis (source: https://upload.wikimedia.org/wikipedia/commons/4/45/Osmosis_computer_simulation.jpg).....	31
Figure 11 - A timeline about e-readers (source: https://mastermindmaps.files.wordpress.com/2013/05/timeline-ebook.jpeg).....	31
Figure 12 - A time series plot of annual number of earthquakes in the world (source: https://onlinecourses.science.psu.edu/stat510/sites/onlinecourses.science.psu.edu/stat510/files/L01/graph_01.gif).....	32

Figure 13 - A Gantt Chart for Completed and Remaining Tasks of Project X (source: <http://www.idealware.org/blog/tools-gantt-charts>).....32

Figure 14 - A Streamgraph about the top 8 name references in Internet by TechCrunch (source: http://www.neoformix.com/2008/sg_techcrunchNames8.png).....32

Figure 15 - An arc diagram (source: http://gastonsanchez.com/images/blog/miserables_arcplot.png).....33

Figure 16 - A Sankey Diagram of Process Energy in 2010 (source: <http://energy.gov/eere/amo/static-sankey-diagram-process-energy-us-manufacturing-sector>).....34

Figure 17 - An Alluvial Diagram in stages (source: <http://mezbaha.tumblr.com>).....34

Figure 18 - Tag Cloud of a University Student (source: <http://libguides.daltonstate.edu/c.php?g=24583&p=148455>).....35

Figure 19 - A tree map about injury mortality per US Census Region (source: https://wiki.cs.umd.edu/cmsc734_f12/images/1/11/Treemap-Regions-Injury.png).36

Figure 20 - Scatter plot showing wife age as a function of husband's age (source: http://onlinestatbook.com/chapter4/graphics/age_scatterplot.gif).....36

Figure 21 - The Bubble Chart shows the number of burglaries versus the number of murders in population (source: <http://glowingpython.blogspot.gr/2011/11/how-to-make-bubble-charts-with.html>).....37

Figure 22 - A five-year global PMI Heatmap (source: <http://www.zerohedge.com/sites/default/files/images/user5/imageroot/2013/07/PMI%20heatmap.jpg>).....37

Figure 23 - Parallel Coordinates for high-dimensional geometry (source: <http://visthis.blogspot.gr/2012/10/assignment-3-parallel-coordinates.html>).....38

Figure 24 - A Box Plot about hours of sleep per day (source: <https://plot.ly/static/img/literacy/boxplot/boxplotfig9.jpg>).....38

Figure 25 - Multiple Line Chart of profit per product type (source: <https://eagereyes.org/wp-content/uploads/2013/04/line-multiple.png>).....39

Figure 26 - This spider chart represents the allocated budget versus actual spending for a given organization (source: https://upload.wikimedia.org/wikipedia/commons/1/18/Spider_Chart2.jpg).....39

Figure 27 - A Dendrogram used for organization (source: https://lh4.googleusercontent.com/oyPtmsae1sjssR53Ya6gxzarbCOcjawcfokYYZ4fv3Z6egjo4_Vh8W-hhuBEIA2jQeVtsijjBepc8iCOfuHVWMj9x9iyVMAe-QaiywMxX_IsutGvyo).....40

Figure 28 - Radial Tree (source: <https://seeingcomplexity.wordpress.com/2011/02/05/hierarchical-edge-bundles>).....41

Figure 29 - The hyperbolic tree of an organization (source: http://www.sigchi.org/chi95/proceedings/papers/jl_bdy.htm).....41

Figure 30 - Double Circular Hierarchy graph (source: <https://seeingcomplexity.wordpress.com/2011/02/05/hierarchical-edge-bundles>).....42

Figure 31 - Node-link diagram example (source: <https://www.syncfusion.com/products/silverlight/images/diagram/silverlight-diagram-radialtreelayout.png>).....42

Figure 32 - The Actual Load of each device of the given houses.....50

Figure 33 - The Bid Price of each device of the given houses.....51

Figure 34 - The Bid Quantity of each device of the given houses.....51

Figure 35 - The result occurred from our parser.....52

Figure 36 - Our Database Formation.....53

Figure 37 - Schematic illustration of MVC Architecture (source: http://easylara.com/wp-content/uploads/2014/11/bb8db_03.jpg).....61

Figure 38 - The index page of GridLab-DVIsor.....63

Figure 39 - The selection form used for Central Triplex Meter table.....64

Figure 40 - The calendar displayed when the user press the Date input.....65

Figure 41 - Multiple Axes Chart for Central Triplex Meter.....65

Figure 42 - Line Diagram for Central Triplex Meter.....66

Figure 43 - Spider Diagram for Central Triplex Meter.....67

Figure 44 - The selection form used for Devices table.....67

Figure 45 - Multiple Axes Diagram for Devices table.....68

Figure 46 - Synchronized Diagrams for Devices table.....69

Figure 47 - The Consumed Load in Heatmap form of the Devices Total Table.....	70
Figure 48 - Multiple Axes Diagram for Energy Sources Table.....	71
Figure 49 - The Line Diagram for Energy Sources table.....	72
Figure 50 - The Spider Diagram for Energy Sources table.....	72
Figure 51 - The selection page for the Houses table.....	73
Figure 52 - The selection of phase in the Houses table.....	74
Figure 53 - The selection of houses in the Houses table.....	75
Figure 54 - Line Diagram for multiple houses of the Houses table.....	76
Figure 55 - Spider Diagram for multiple houses of the Houses table.....	77
Figure 56 - The selection page for Market Pool table.....	78
Figure 57 - The Line Diagram for Market Pool table.....	79
Figure 58 - The Spider Diagram for Market Pool table.....	79
Figure 59 - The selection page of Nodes table.....	80
Figure 60 - Hierarchical Layout of the Nodes table.....	81
Figure 61 - Unsorted Graph of the Nodes table.....	81
Figure 62 - The Hierarchical Layout of phase A of the Nodes table.....	82
Figure 63 - The charts per phase in the Transformer's table.....	83
Figure 64 – The System Graph of our application.....	84

Chapter 1

1. Introduction

1.1. Introduction to Big Data and Data Visualization Techniques

Recent years have seen a large increase in data we receive daily because of the development of technology and the Internet. The development of technology contributes to an increased daily production of data, while the development of the Internet contributes to sharing these data in order to reach as many users as possible. For users to succeed in filtering and consuming only the data that really matter to them, appropriate visualization techniques are needed.

Visualization techniques produce (interactive) graphic visual representations of abstract data to present information quickly and clearly and reinforce human cognition; thus enabling the viewer to gain knowledge about the internal structure of the data and causal relationships in it.

Because, as we already said, we are talking about large and complex data – the term used for this kind of data is Big Data and we will use this term in our analysis – traditional visualization techniques are not enough to improve human's cognition. Modern visualization techniques may enhance the human visual system's ability to see patterns and trends.

In this thesis project we implement some visualization techniques to data that we receive from a power distribution system simulation called GridLAB-D. In simulation system GridLAB-D we executed some experiments and we use the data it produced as the basis of our application. Essentially we will attempt to visualize GridLAB-D produced data in the clearest possible way.

1.2. About this diploma thesis project

In this thesis project we are trying to design and develop an application based on the web which draws data from the GridLAB-D system and present them to the user in a way that they can perceive them easily and directly using their sight. To be more specific we receive valuable, but raw data from a power distribution system simulation (GridLAB-D) and we are trying to convert them with **data visualization** techniques, so this interaction between the user and the application should result in the production of knowledge. This application uses JavaScript libraries to visualize the raw data received from the simulation system to achieve knowledge for the user.

Our goal is to use this information to visually present the energy profile of the given system, like the energy profile of each node and house of the system, or the energy profile of each device belonging to the system, as well as more general tables, such as the market pool of the whole system, which shows how the values of price and quantity fluctuate with each passing day.

1.3. Content's organization

This thesis project is structured in the following way:

In **chapter 2**, we describe the theory and fundamentals of data visualization. Firstly, we make a small introduction to the huge amounts of data and information of the web, which is what we call Big Data. Secondly, we describe in depth the fundamentals of the data visualization theory. Finally, we present some of the most widely used tools and software that exist today, and are used in data visualization.

In **chapter 3**, we describe how we converted the raw data given to us to a specific, needed format and how we organized and prepared them for insertion to our database. Lastly, we analyze the structure and schema of our database.

In **chapter 4**, we describe the architecture we used to develop our web application. We also describe all the tools and technologies we used to design and develop our web application. Finally, we analyze the user interface we created.

In **chapter 5**, we describe some of the most important differences between our implementation and existing programs.

Finally in **chapter 6**, we describe our conclusions and discuss possible future steps.

Chapter 2

2. Data Visualization

2.1. What is Big Data?

In recent years we see an increase in the amount of daily data we receive, such as polls, web page analytics and stock indices. With the wide spread of technology and the Internet, scientists observed that every day consistently more data are produced. As the Internet gains in popularity, the amount of data grows. Data grow in size in part because they are increasingly being gathered by cheap and numerous information – sensing mobile devices, aerial (remote sensing), software logs, cameras, microphones, wireless sensor networks, and radio-frequency identification (RFID) readers. And thereby data have begun to surge and a new term has been born: Big Data. Big Data is a popular term used to describe the exponential growth and availability of data, both structured and unstructured. And big data may be as important to business – and society – as the Internet has become. Why? More data may lead to more accurate analysis. And more accurate analysis may lead to more confident decision making. And better decisions can mean greater operational efficiencies, cost reductions and reduced risk. So, the Big Data can be described from the following characteristics:

Volume The quantity of generated data is important in this context. The size of the data determines the value and potential of the data under consideration, and whether it can actually be considered big data or not. The name ‘big data’ itself contains a term related to size, and hence the characteristic.

Variety The type of content, and an essential fact that data analysts must know. This helps people who are associated with and analyze the data to effectively use the data to their advantage and thus uphold its importance.

Velocity In this context, the speed at which the data is generated and processed to meet the demands and the challenges that lie in the path of growth and development.

Variability The inconsistency the data can show at times— which can hamper the process of handling and managing the data effectively.

Veracity The quality of captured data, which can vary greatly. Accurate analysis depends on the veracity of source data.

Complexity Data management can be very complex, especially when large volumes of data come from multiple sources. Data must be linked, connected, and correlated so users can grasp the information the data is supposed to convey. [1]

So a question arises; how can the human mind grasp all the data it receives daily, easily, quickly and without causing headaches? The answer to this question is Data Visualization.

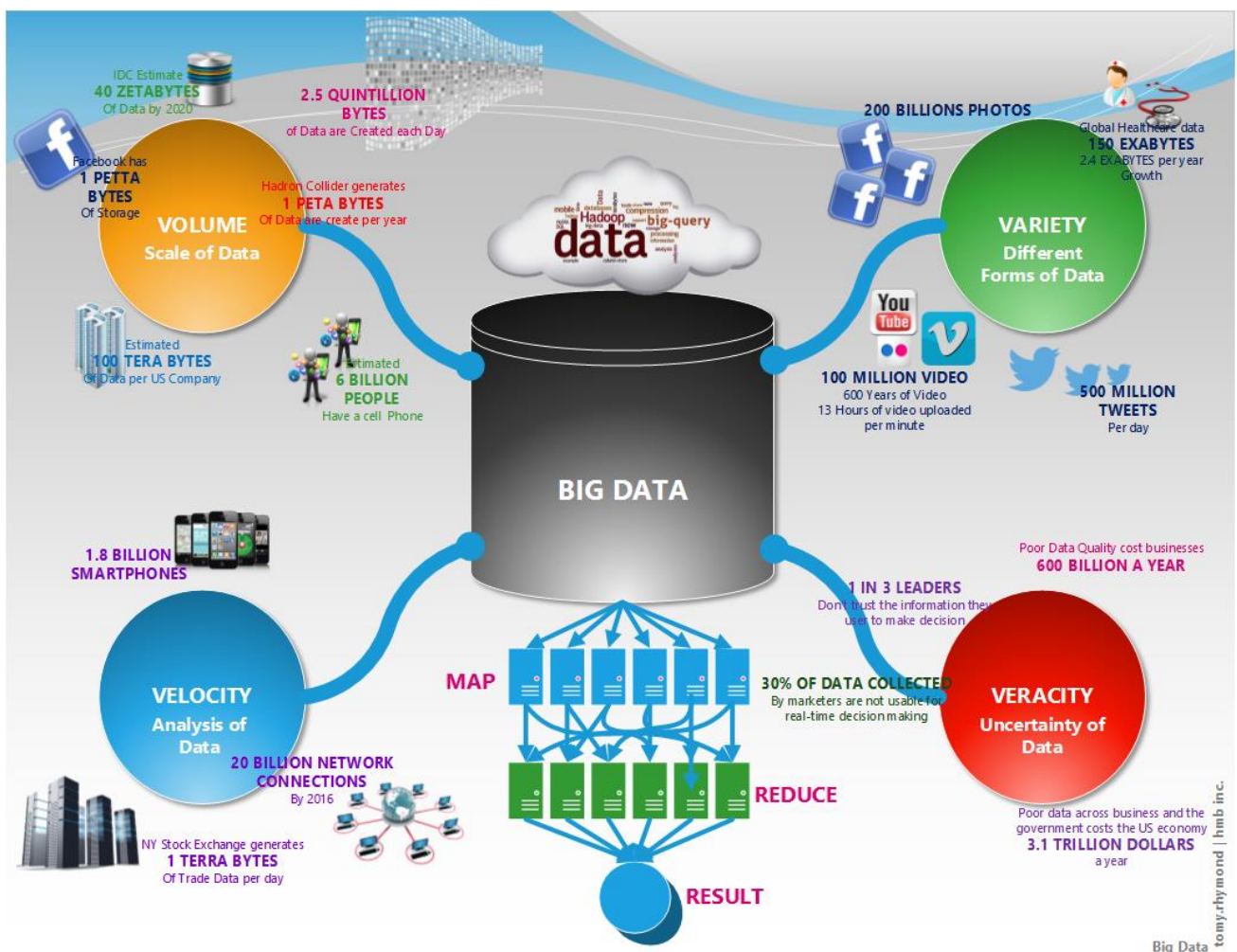


Figure 1 An infographic about the four v's of Big Data.

2.2. What is Data Visualization?

Data Visualization is a general term that describes any effort to help people understand the significance of data by placing it in a visual context. Visualizations help people see things that were not obvious to them before. Even when data volumes are very large, patterns can be spotted quickly and easily. Visualizations convey information in a universal manner and make it simple to share ideas with others. So, Data Visualization is the presentation of abstract data in a pictorial or graphical format, a presentation which must follow specific design principles that are derived from an understanding of human perception. These principles have to be based on categories that human mind can understand. According to Russell Ackoff, the content of the human mind can be classified into five categories:

Data with this term we mean raw data. It simply exists and has no significance beyond its existence. It does not give the user any use of its existence.

Information is data that has been given meaning by way of relational connection. Provides answers to “who”, “what”, “where”, and “when” questions.

Knowledge is the application of data and information. This term answers “how” questions.

Understanding is the appreciation of “why”. Some people are going to ask themselves what is the difference between knowledge and understanding. So, the difference between knowledge and understanding is the difference between “memorizing” and “learning”.

Wisdom incorporates vision and design. It is the only of the five categories that deals with the future. Wisdom is actually the evaluation of understanding. [2, 3]

2.3. Visualization Techniques

The increased raw data we receive daily, poses the need of proper visualization so that we may "read" them. Besides, the saying "a picture is worth a thousand words" is not accidental. Studies show that the average human brain processes images 60,000 times faster than text. So, what we need to do is find proper visualization techniques that help us convert raw data into a form easily conceived by the user.

There are many factors to take into consideration in order to design a visualization.

In this paper we will separate the graphs in categories that basically depend on the number of dimensions that we want to include and secondarily based on features like time and clustering. The main visualization techniques of each category are following.

2.3.1. 1 – Dimension or Linear

When we talk about one-dimensional charts we usually mean lists of data items, organized by a single feature (e.g. alphabetical order or order by value).

2.3.2. 2 – Dimension or Planar

This data visualization category is basically appropriate for geospatial data. The most famous techniques of this category are the following:

2.3.2.1. Choropleth Map

A choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map, such as population density and order-capital income. The choropleth map provides an easy way to visualize how a measurement varies across a geographic area or it shows the level of variability within a region. [1]

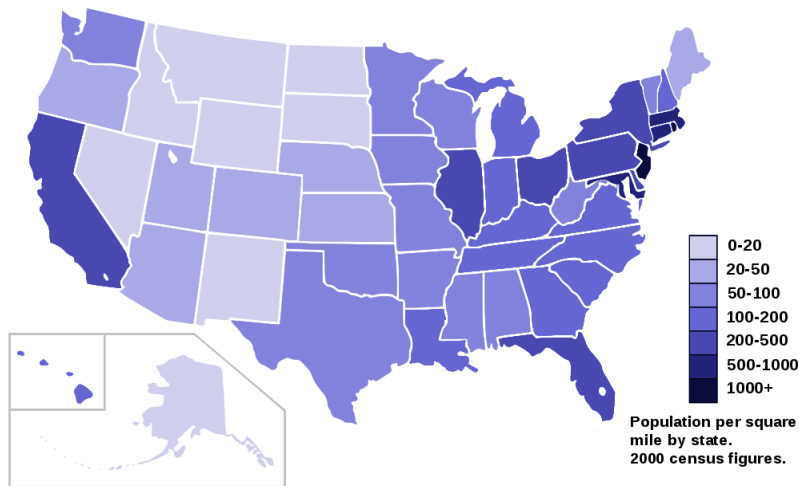


Figure 2 A choropleth map that visualizes the population per square mile by state.

2.3.2.2. Dasymetric Map

A dasymetric map is an alternative to a choropleth map. As with a choropleth map, data are collected by enumeration units. But instead of mapping the data so that the region appears uniform, *ancillary information* is used to model internal distribution of the phenomenon. For example, population density will be much lower in forested area than urbanized area, so in a common operation, land cover data (forest, water, grassland, urbanization) may be used to model the distribution of population reported by census enumeration unit such as a tract or county. [1]

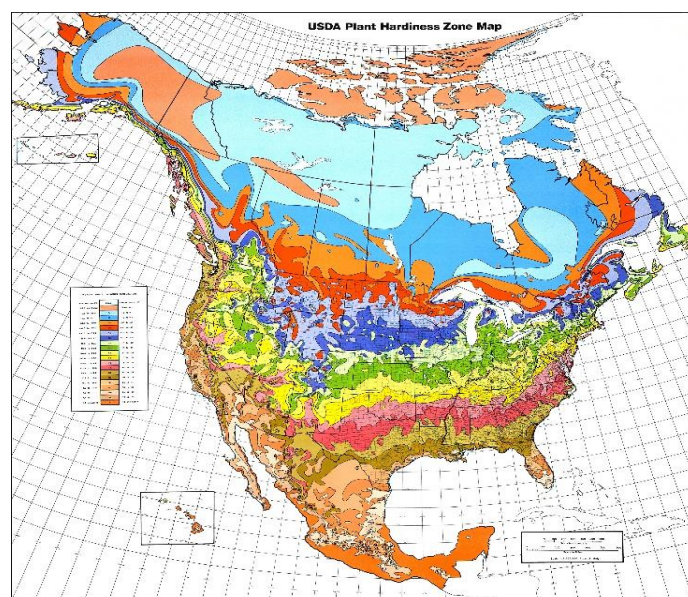


Figure 3 Dasymetric map of climate and plant hardiness zones.

2.3.2.3. Cartogram

A cartogram is a map in which some thematic mapping variable – such as travel time, population, or Gross National Product – is substituted for land area or distance. The geometry or space of the map is distorted in order to convey the information of this alternate variable. [1]

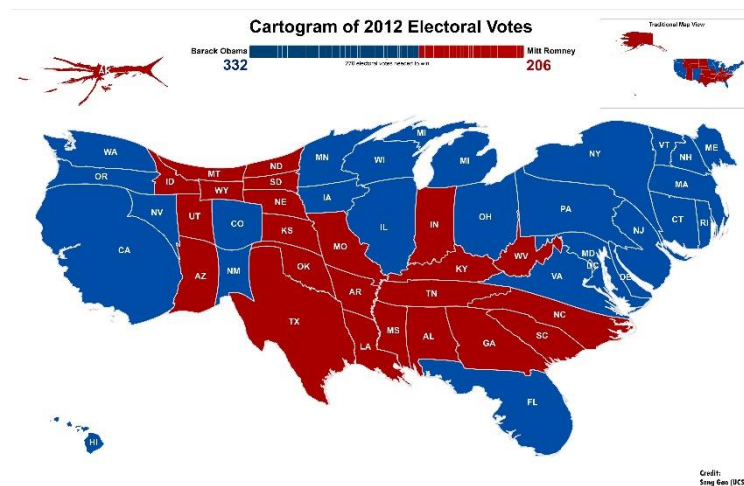


Figure 4 A cartogram of 2012 electoral votes.

2.3.2.4. Dot Distribution Map

A dot distribution map (also known as *dot density map*) is as a map type that uses a dot symbol to show the presence of a feature or phenomenon. Dot maps rely on a visual scatter to show spatial pattern. There are two sub-categories of this visualization pattern. Firstly, the “one-to-one” dot map where each dot represents one single recording of a phenomenon and secondly the “one-to-many” (or dot density map) where each dot on the map represents more than one of the phenomena being mapped. [1]

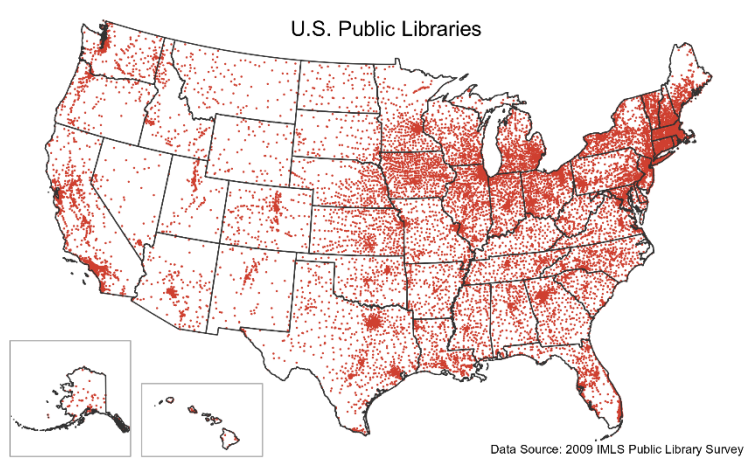


Figure 5 A one-to-one dot map of U.S. Public Libraries.

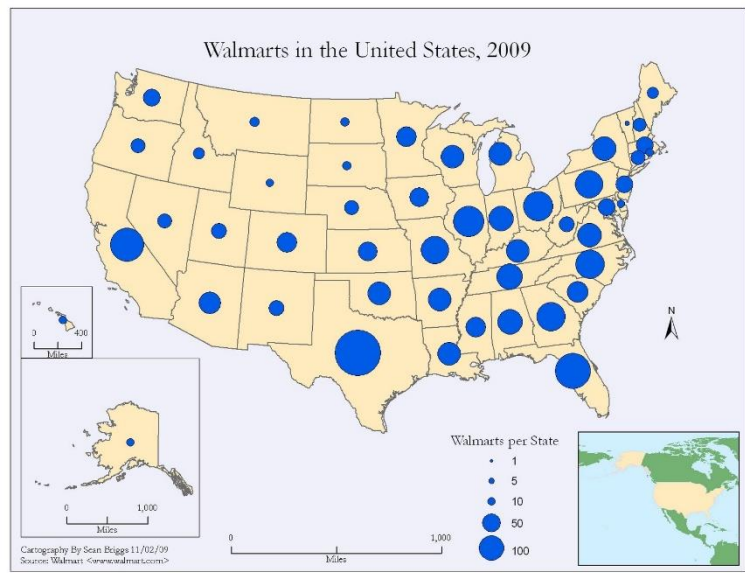


Figure 6 A one-to-many dot map of Walmarts in the United States in 2009.

2.3.2.5. Proportional Symbol

The proportional symbol technique uses symbols of different sizes to represent data associated with different areas or locations within the map. For example, a bottle of wine may be shown at the location of each country in a map, with the area of the bottle being proportional to the population of the city. [1]

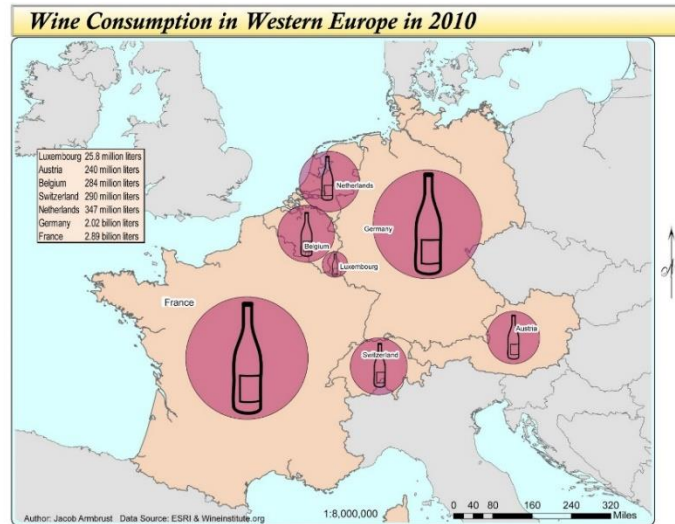


Figure 7 A proportional symbol map of wine consumption in Western Europe in 2010.

2.3.2.6. Contour/ Isopleth/ Isarithmic Map

Contour maps, also known as isopleth or isarithmic maps depict smooth continuous phenomena such as precipitation or elevation. Each line-bounded area on this type of map represents a region with the same value. For example, on an elevation map, each elevation line indicates an area at the listed elevation. An Isarithmic map is a planimetric graphic representation of a 3-D surface. Isarithmic mapping requires 3-D thinking for surfaces that vary spatially. [1]

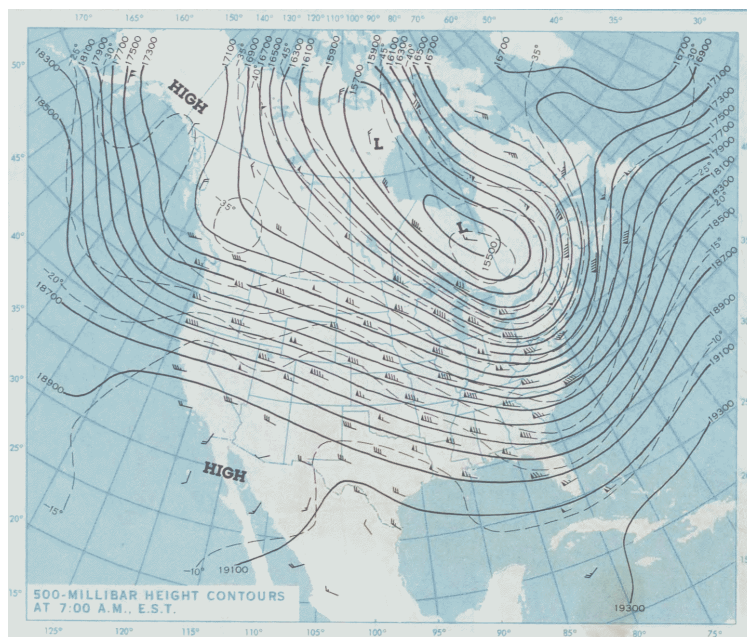


Figure 8 A contour map of 500 millibar height in 1982.

2.3.3. 3 – Dimension or Volumetric

This data visualization category is basically used for scientific visualization. The most famous techniques of this category are the following:

2.3.3.1. 3D Modeling

In 3D computer graphics, 3D modeling is the process of developing a mathematical representation of any three-dimensional surface of an object (either inanimate or living) via specialized software. [1]

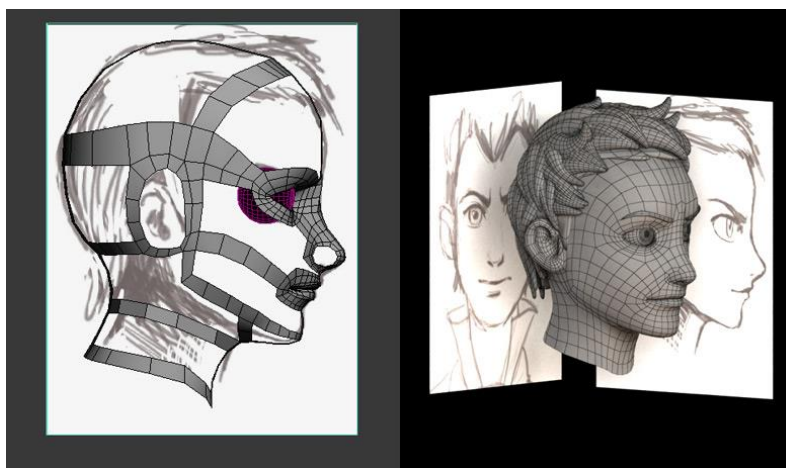


Figure 9 A 3D computer graphics model.

2.3.3.2. Computer Simulation

Computer simulation is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system. Computer simulations have become a useful part of mathematical modelling of many natural systems in physics, and computational physics, chemistry and biology; human systems in economics, psychology, and social science; and in the process of engineering and new technology, to gain insight into the operation of those systems, or to observe their behavior. [1]

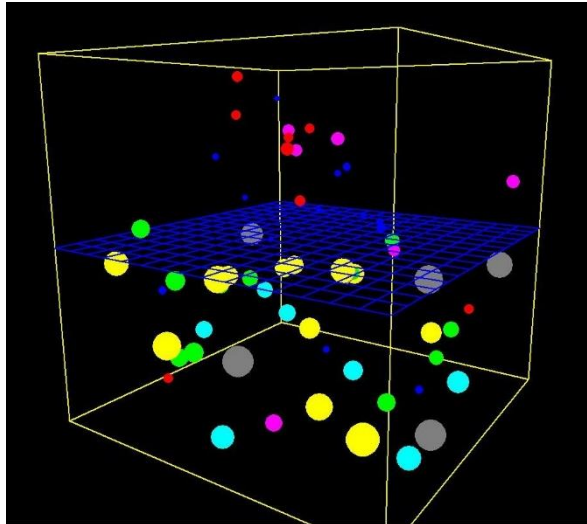


Figure 10 Computer simulation of the process of osmosis.

2.3.4. Temporal

In this data visualization category we represent events according to chronological order. The most famous techniques of this category are the following:

2.3.4.1. Timeline

A timeline is a way of displaying a list of events in chronological order, sometimes described as a project artifact. [1]

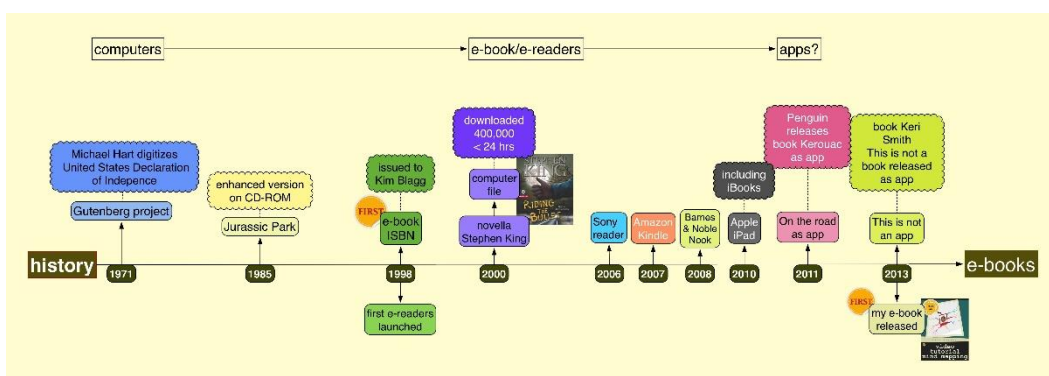


Figure 11 A timeline about e-readers.

2.3.4.2. Time Series

A time series is a sequence of data points, typically consisting of successive measurements made over a time interval. [1]

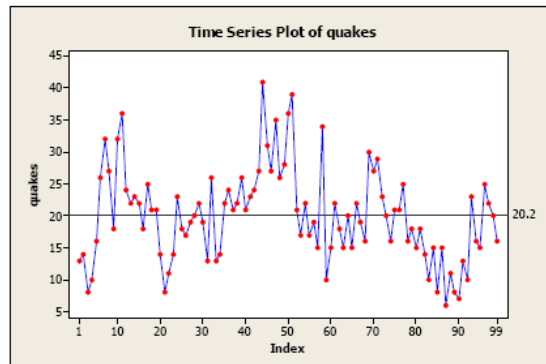


Figure 12 A time series plot of annual number of earthquakes in the world.

2.3.4.3. Gantt Chart

Gantt chart is a type of bar chart that illustrates a project schedule. It illustrates the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. [1]

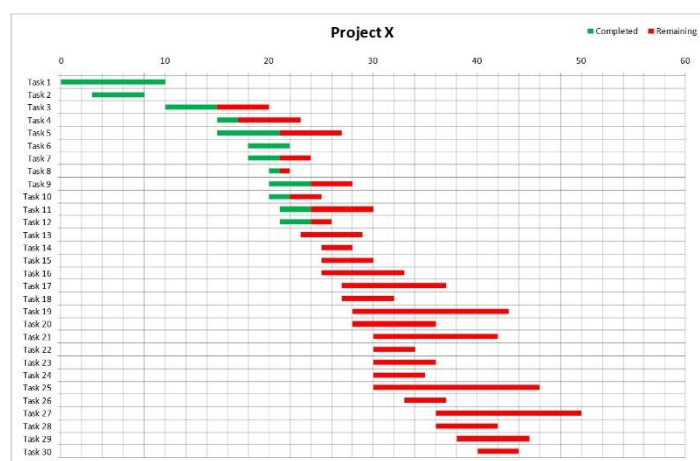


Figure 13 A Gantt Chart for Completed and Remaining Tasks of Project X.

2.3.4.4. Streamgraph

A streamgraph, or stream graph, is a type of stacked area graph which is displaced around a central axis, resulting in a flowing, organic shape. [1]

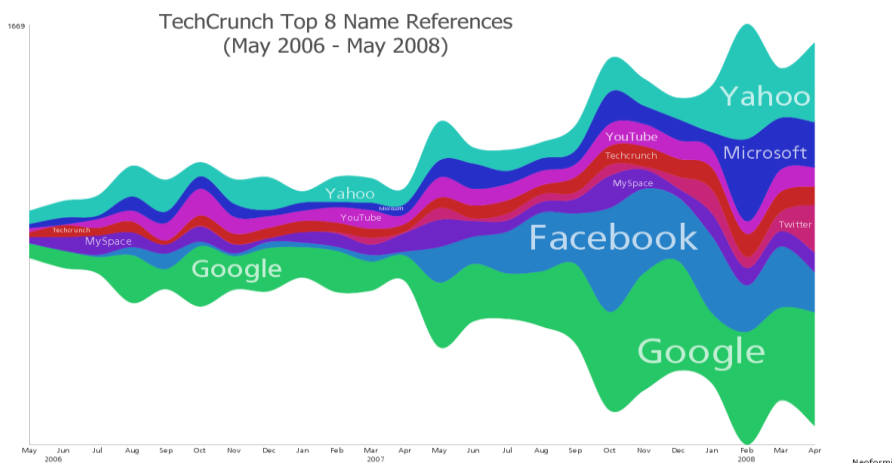


Figure 14 A Streamgraph about the top 8 name references in Internet by TechCrunch.

2.3.4.5. Arc Diagram

An arc diagram is a style of graph drawing, in which the vertices of a graph are placed along a line in the Euclidean plane, with edges being drawn as semicircles in one of the two half planes bounded by the line, or as smooth curves formed by sequences of semicircles. [1]

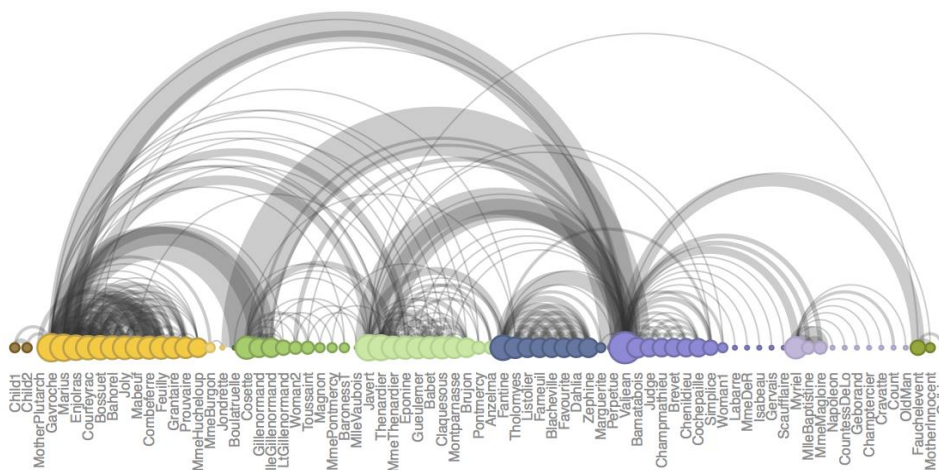


Figure 15 An arc diagram.

2.3.4.6. Sankey Diagram

Sankey diagrams are a specific type of flow diagram, in which the width of the arrows is shown proportionally to the flow quantity. [1]

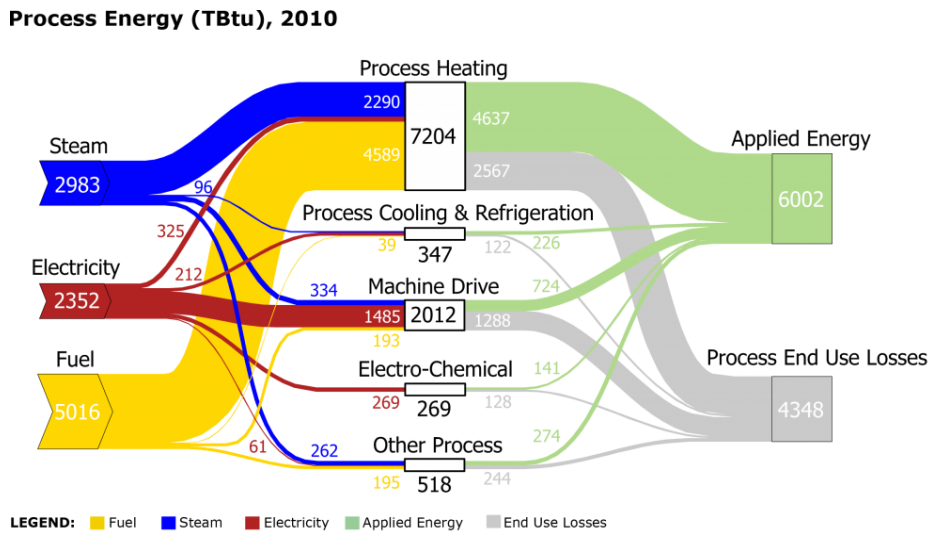


Figure 16 A Sankey Diagram of Process Energy in 2010.

2.3.4.7. Alluvial Diagram

Alluvial diagrams are a type of flow diagram originally developed to represent changes in network structure over time. In allusion to both their visual appearance and their emphasis on flow. [1]

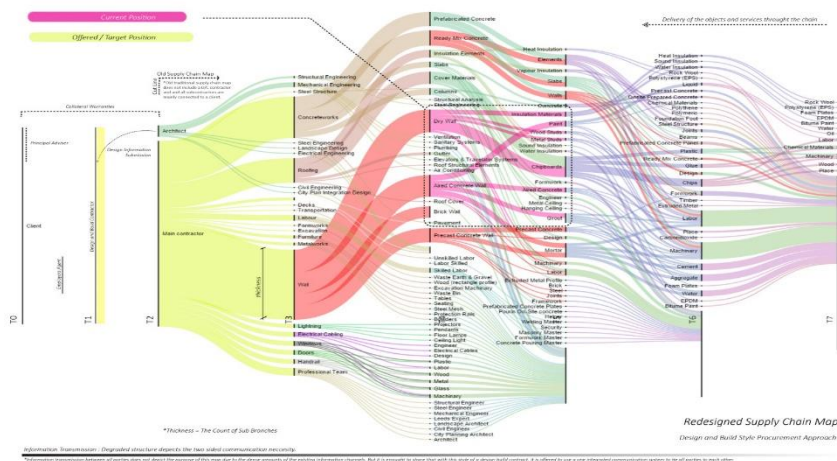


Figure 17 An Alluvial Diagram in stages.

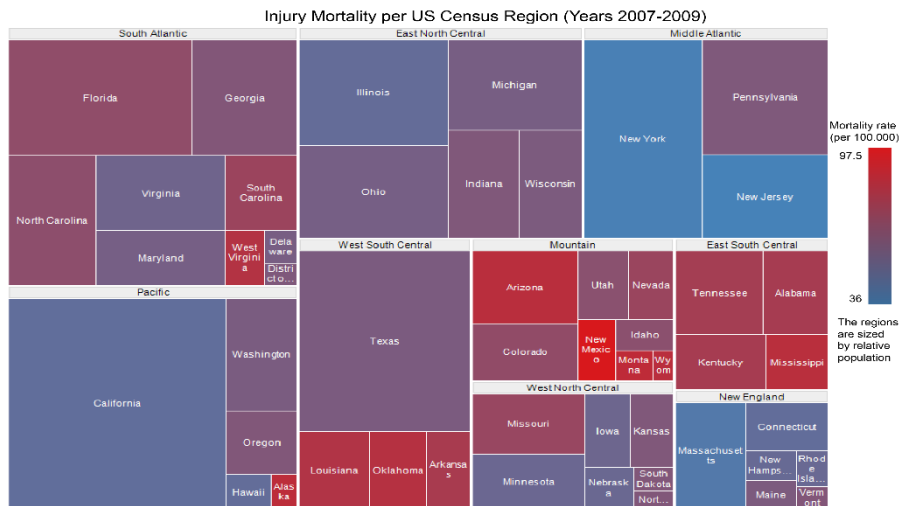


Figure 19 A tree map about injury mortality per US Census Region.

2.3.5.3. Scatter Plot

A scatter plot is a type of mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data. If the points are color-coded you can increase the number of displayed variables to three. [1]

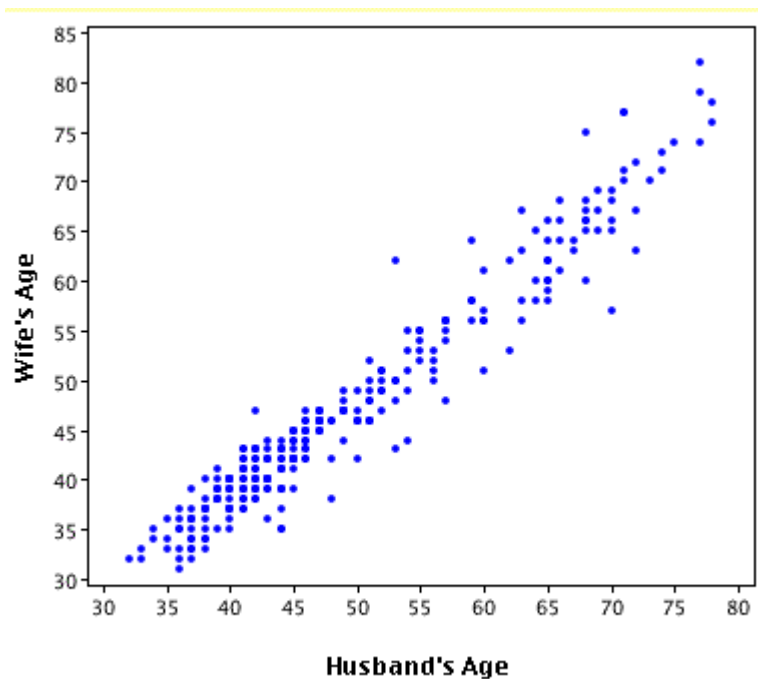


Figure 20 Scatter plot showing wife age as a function of husband's age.

2.3.5.4. Bubble Chart

A bubble chart is a type of chart that displays three dimensions of data. Bubble charts can facilitate the understanding of social, economical, medical, and other scientific relationships. [1]

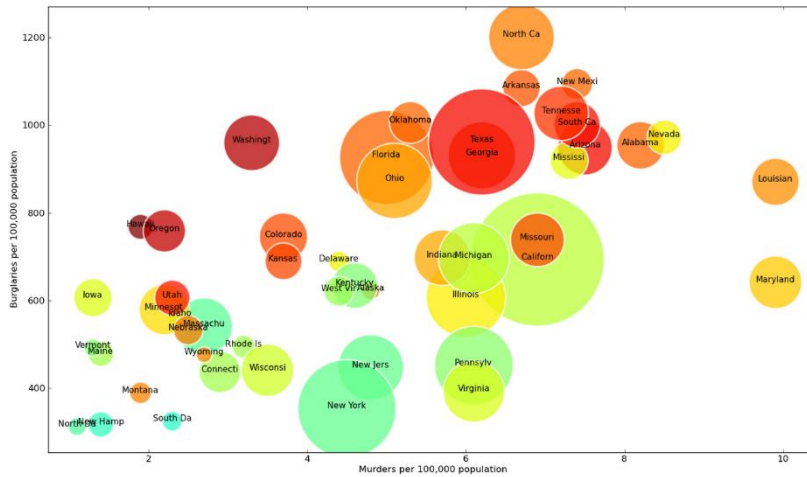


Figure 21 The Bubble Chart shows the number of burglaries versus the number of murders in population.

2.3.5.5. Heatmap

A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors. Fractal maps and tree maps both often use a similar system of color-coding to represent the values taken by a variable in a hierarchy. [1]

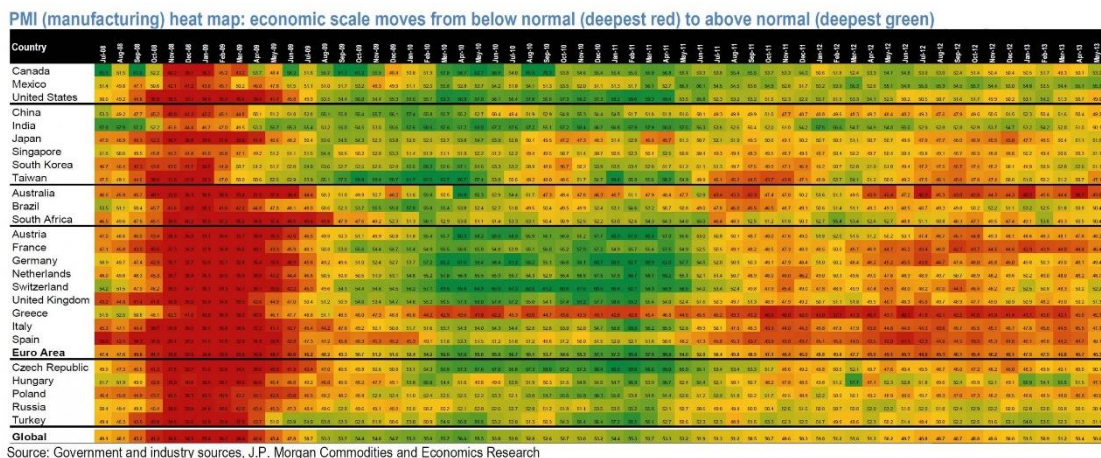


Figure 22 A five-year global PMI Heatmap.

2.3.5.6. Parallel Coordinates

Parallel coordinates is a common way of visualizing high-dimensional geometry and analyzing multivariate data. [1]

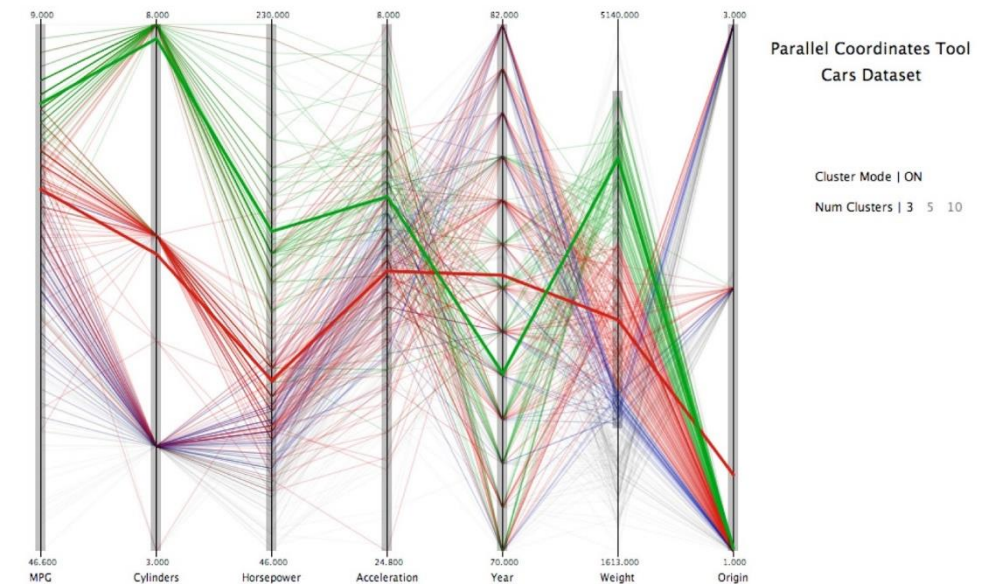


Figure 23 Parallel Coordinates for high-dimensional geometry.

2.3.5.7. Box Plot

In descriptive statistics, a box plot or boxplot is a convenient way of graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (*whiskers*) indicating variability outside the upper and lower quartiles. [1]

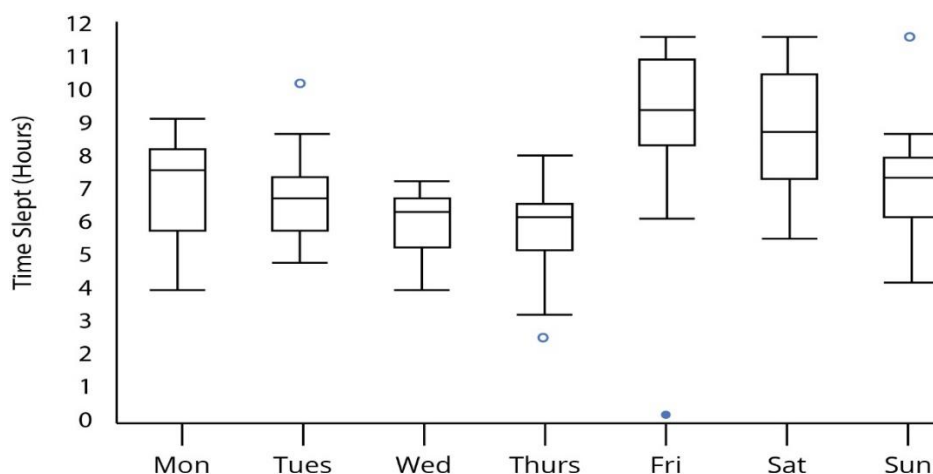


Figure 24 A Box Plot about hours of sleep per day.

2.3.5.8. Line Diagram

A line chart is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments. It is a basic type of chart common in many fields. [1]

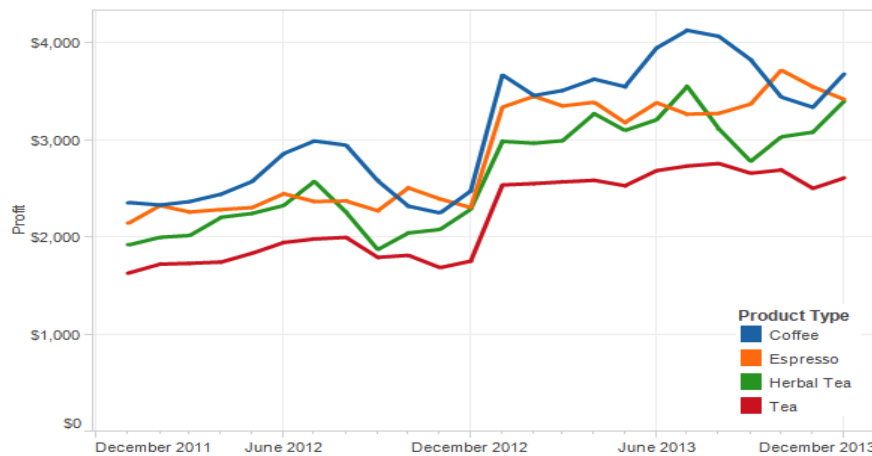


Figure 25 Multiple Line Chart of profit per product type.

2.3.5.9. Radar/ Spider Diagram

A radar chart is a graphical method of displaying multivariate data in the form of a two-dimensional chart of three or more quantitative variables represented on axes starting from the same point. The relative position and angle of the axes is typically uninformative. [1]

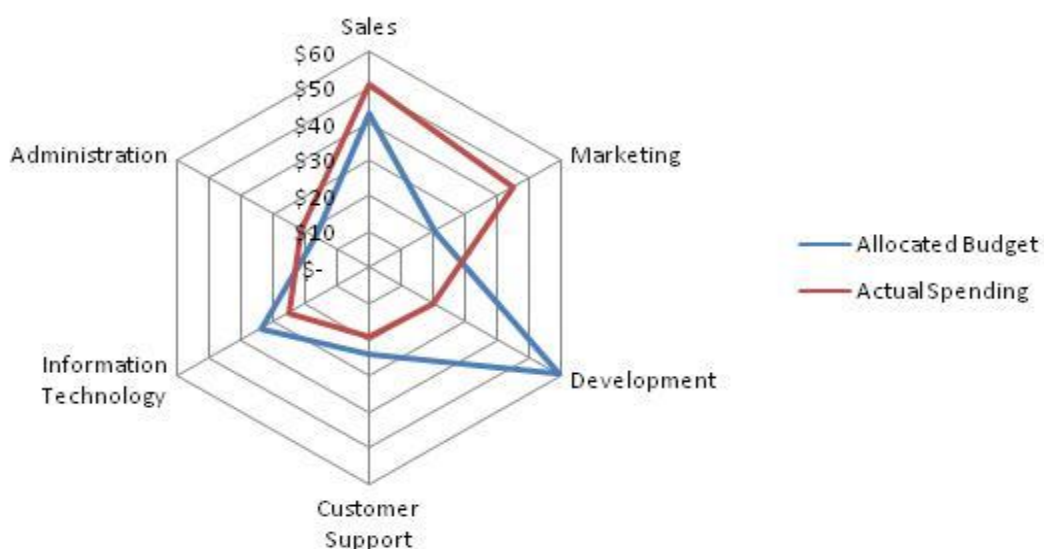


Figure 26 This spider chart represents the allocated budget versus actual spending for a given organization.

2.3.6. Tree/ Hierarchical

This data visualization category associates with clustering. The most famous techniques of this category are the following:

2.3.6.1. Dendrogram

A dendrogram is a tree diagram frequently used to illustrate the arrangement of the clusters produced by hierarchical clustering. Dendrograms are often used in computational biology to illustrate the clustering of genes or samples. [1]

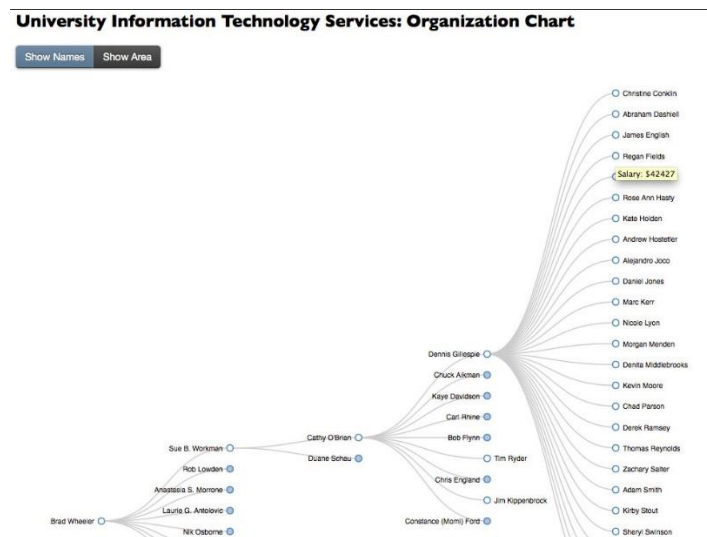


Figure 27 A Dendrogram used for organization.

2.3.6.2. Radial Tree

A radial tree, is a method of displaying a tree structure (e.g., a tree data structure) in a way that expands outwards, radially. It is one of many ways to visually display a tree, with examples extending back to the early 20th century. In use, it is a type of information graphic. [1]

2.3.7. Networks

This data visualization category associates with clustering. The most famous techniques of this category are the following:

2.3.7.1. Circular Hierarchy/ Dependency Graph

A dependency graph is a directed graph representing dependencies of several objects towards each other. It is possible to derive an evaluation order or the absence of an evaluation order that respects the given dependencies from the dependency graph. [1]

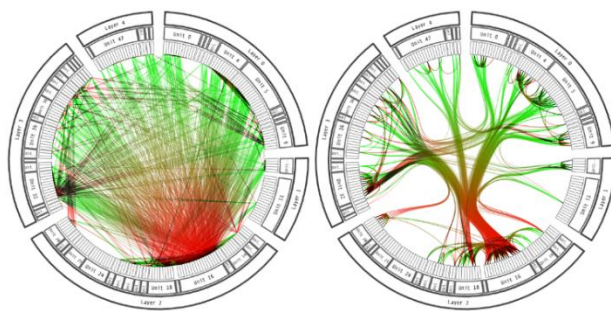


Figure 30 Double Circular Hierarchy graph.

2.3.7.2. Node-link Diagram

A node-link diagram is a type of data visualization that captures entities as nodes (vertices) and relationships (links). A node is represented by a circle, and a link is represented by a line.

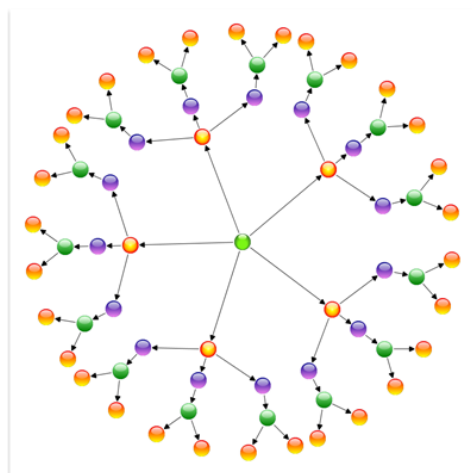


Figure 31 Node-link diagram example.

2.4. Data Visualization Tools

Raw data are boring and difficult to make sense of in their natural form. Add visualization and you get something that everybody can easily digest. Not only you can make sense of it faster, but you can also observe interesting patterns that wouldn't be apparent just by looking only at stats. To achieve the best possible visualization, we usually use either technologies for graphics inside the web browser, like SVG and HTML 5 Canvas, or JavaScript Libraries to add functionality and make our web page more appealing.

2.4.1. SVG & HTML 5 Canvas

2.4.1.1. SVG

Scalable Vector Graphics (SVG) is an XML-based vector image format for two-dimensional graphics with support for interactivity and animation. The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999.

SVG images and their behaviors are defined in XML text files. This means that they can be searched, indexed, scripted, and compressed. As XML files, SVG images can be created and edited with any text editor, but are more often created with drawing software. In contrast to JPEG and GIF images on the Web, which are bitmapped and always remain a specified size, SVG images are scalable to the size of the viewing window and will adjust in size and resolution according to the window in which it is displayed. [1]

2.4.1.2. HTML 5 Canvas

The canvas element is part of HTML5 and allows for dynamic, scriptable rendering of 2D shapes and bitmap images. HTML5 canvas gives you an easy and powerful way to draw graphics using JavaScript. For each canvas element you can use a "context" (think about a page in a drawing pad), into which you can issue JavaScript commands to draw anything you want. Browsers can implement multiple canvas contexts and the different APIs provide the drawing functionality. [1]

2.4.2. JavaScript Libraries

2.4.2.1. General Libraries

2.4.2.1.1. D3.js

D3.js, short for 'Data Driven Documents', is the first name that comes to mind when we think of Data Visualization. It uses HTML, CSS, and SVG to render some amazing charts and diagrams. If you can imagine any visualization, you can do it with D3. Also, D3 emphasize on web standards giving full capabilities of modern browsers and combining powerful visualization components and a data-driven approach to DOM manipulation. It is feature packed, interactivity rich and extremely beautiful. Most of all it's free and open-source.

2.4.2.1.2. Highcharts

Highcharts is another big player in the charting space. It also offers a diverse range of charts and maps right out of the box. Other than normal charts, it also offers a different package for stock charts called Highstock which is also feature rich.

It allows exporting charts in PNG, JPG, SVG and PDF. Highcharts is free for non-commercial and personal use. It is 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. It works in all modern mobile and desktop browsers. We have used Highcharts in some of our chart visualizations in our project.

2.4.2.1.3. FusionCharts

FusionCharts has probably the most exhaustive collection of charts and maps. With over 90+ chart types and 965 maps, you'll find everything that you need right out of the box. It not only supports modern browsers, but also older browsers starting from IE 6.

FusionCharts supports both JSON and XML data formats, and you can export charts in PNG, JPEG, SVG or PDF. They have a nice collection of business dashboards and live demos for inspiration.

Their charts and maps work across all devices and platforms, are highly customizable and have beautiful interactions. One thing to keep in mind about FusionCharts is that it's slightly expensive. But you can always get started with their unrestricted free trial and then buy if you like it.

2.4.2.1.4. Charts.js

Chart.js is a tiny open source library that supports just six chart types: line, bar, radar, polar, pie and doughnut. But the reason I like it is that sometimes that's all the charts one needs for a project. If the application is big and complex, then libraries like Google Charts and FusionCharts makes sense, otherwise for small hobby projects Chart.js is the perfect solution. It uses HTML5 canvas element for rendering charts. All the charts are responsive and use flat design. It is one of the most popular open-source charting libraries to emerge recently.

2.4.2.1.5. Google Charts

Google Charts renders charts in HTML5/SVG to provide cross-browser compatibility and cross-platform portability to iPhones and Android. It also includes VML for supporting older IE versions. It offers a decent number of charts which covers the most commonly used chart types like bar, area, pie and gauges. It is flexible and user friendly.

2.4.2.1.6. Raphaël

Raphaël is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library. It uses SVG and VML as a base for creating graphics. This means every graphical object you create is also a DOM object, so you can attach JavaScript event handlers or modify them later. Raphaël's goal is to provide an adapter that will make drawing vector art compatible cross-browser and easy. [4]

2.4.2.2. Graph visualization libraries

2.4.2.2.1. Vis.js

Vis.js is a dynamic, browser based visualization library. It is designed to be easy to use, to handle large amount of dynamic data, and to enable manipulation of and interaction with the data. The library consists of the components DataSet, Timeline, Network, Graph2d and Graph3d. In our project we have used Vis.js in some of our visualizations. [5]

2.4.2.2.2. Graph Dracula

Dracula is a set of tools to display and layout interactive graphs, along with various related algorithms. No Flash, no Java, no plug-ins. Just plain JavaScript and SVG. The code is released under the MIT license, so commercial use is not a problem. Creating a graph is simple! You also can customize anything easily. [6]

2.4.2.2.3. jsPlumb

jsPlumb is a jQuery plug-in for creating interactive connected graphs. It has everything you need to build an application, such as pan/ zoom, a minimap widget, automatic layouts and data binding. It uses HTML5 and CSS3 and it has seamless integration with mobile devices. [7]

2.4.2.2.4. Sigma.js

Sigma is a JavaScript library dedicated to graph drawing. It is an open-source lightweight library to make network visualizations using the HTML canvas element. [8]

2.4.2.2.5. Cytoscape.js

Cytoscape.js is an open-source network library written in JavaScript. You can use Cytoscape.js for graph analysis and visualization. It allows you to easily display and manipulate rich, interactive graphs. Cytoscape.js supports both desktop and mobile

browsers. Cytoscape.js includes all the gestures you would expect out-of-the-box, including pinch-to-zoom, box selection, panning and much more. [9]

2.4.2.3. JavaScript Libraries for Maps

2.4.2.3.1. Leaflet

Leaflet is an open-source library developed for mobile-friendly interactive maps. It is extremely light and has lots of features for making any kind of maps. It uses HTML5 and CSS3 for rendering maps, and works across all major desktop and mobile platforms. There is a wide range of plugins available for adding features like animated markers, heatmaps etc. that extend the core functionality.

2.4.2.3.2. Polymaps

Polymaps is a free JavaScript library for making dynamic, interactive maps in modern web browsers. Polymaps uses SVG to draw the maps. It provides speedy display of multi-zoom datasets over maps, and supports a variety of visual presentations for tiled vector data. [10]

2.4.2.3.3. OpenLayers

OpenLayers is a high performance open source JavaScript framework to build interactive maps using various mapping services. You can choose the map layer source using tiled layer or vector layer from a number of map services. [11]

2.4.2.3.4. Kartograph

Kartograph is a simple and lightweight framework for building interactive map applications without Google Maps or any other mapping service. [12]

2.4.2.3.5. jHERE

jHERE is a powerful map API with which you can easily add interactive maps to your site. It is simple and lightweight. [13]

2.4.2.4. Time Series Visualization Libraries

2.4.2.4.1. MetricsGraphics.js

MetricsGraphics.js is a library built on top of D3 that is optimized for visualizing and laying out time-series data. It provides a simple way to produce common types of graphics in a principled, consistent and responsive way. [14]

2.4.2.4.2. Cubism.js

Cubism.js is another D3 plugin for visualizing time series. Use Cubism to construct better realtime dashboards, pulling data from Graphite, Cube and other sources. Cubism is available under the Apache License on GitHub. [15]

2.4.2.4.3. Crossfilter

Crossfilter is a JavaScript library for exploring large multivariate datasets in the browser. Crossfilter supports extremely fast (<30ms) interaction with coordinated views, even with datasets containing a million or more records. [16]

2.4.2.4.4. Rickshaw

Rickshaw is a JavaScript toolkit for creating interactive time series graphs. It's all based on D3 underneath, so graphs are drawn with standard SVG and styled with CSS. [17]

Chapter 3

3. System Design

Our system is based on the MVC architecture, which means that each logic layer of the system is isolated from the other. MVC is the abbreviation of Model – View – Controller, where each one of these terms is a separate layer of our system. The Model Layer is connected with our application’s database and it is responsible for storing and loading data to/from it. The View Layer is responsible for displaying all or a portion of the data to the user. And, finally, the Controller Layer is the software code that controls the interactions between the Model and View. We will talk more extensively about the MVC architecture subsequently in our project. In this chapter we will analyze the goals that we wanted to achieve with this web system, given the data available to us. We will also analyze the design of our system and reasoning behind this design. So, in this chapter, we will actually be talking about the Model Layer of MVC architecture.

3.1. Organizing our Data

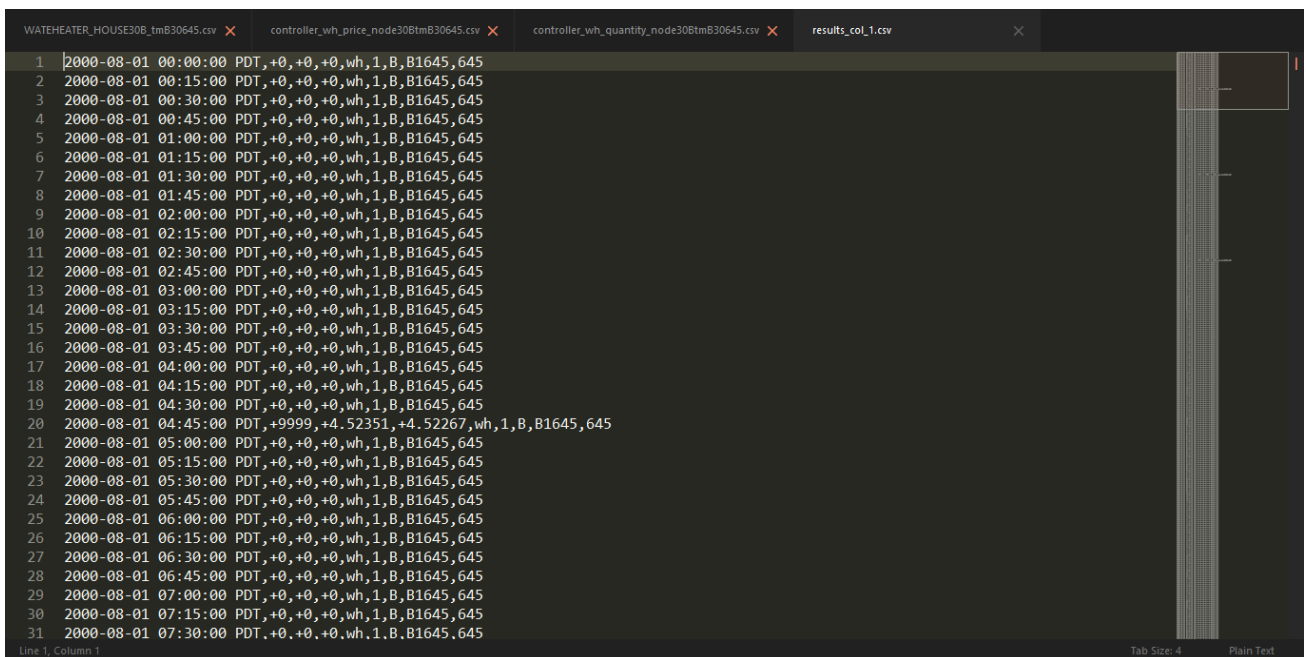
As we mentioned above, the goal of this thesis project is to represent the data we obtained such that it is easily understood by the users of the application. To achieve this we used some visualization techniques. But before that, we had to organize our data in a way that allows us to use any form of visualization later.

To achieve our goals we rely on primary data, which are retrieved from experiments based on the simulation program GridLAB-D with help of PhD student Antonia Nasiakou.

However, data given to us were raw and could not be inserted to our database in their original form due to inconsistencies between their format and the format dictated by our database. Thus, some kind of reformatting should take place, converting all original data to a desired format, while keeping their overall size under a certain threshold to help speed up querying the database.

For this reason we created parsers that take raw data provided by the experiments as their input and output them in a form able to be imported to our database. The parsers we created were developed in JavaScript as a Node.js application. To develop them, we used JavaScript libraries, like underscore.js and string.js to manage and modify the

Our goal was to create a new CSV file, in which each row would be unique and would contain the timestamp of the record, the unique id of the device, the id of the house in which the device belongs, the id of the node in which the house belong, the phase of the house, the actual load, the bid price and the bid quantity of each device. In other words, we would like to create a CSV file in which each unique can be inserted in our database. In Figure 35, we can see the format of the CSV file we created with our parser.



```
WATEHEATER_HOUSE30B_tmB30645.csv | controller_wh_price_node30BtmB30645.csv | controller_wh_quantity_node30BtmB30645.csv | results_col_1.csv
1 | 2000-08-01 00:00:00 PDT,+0,+0,wh,1,B,B1645,645
2 | 2000-08-01 00:15:00 PDT,+0,+0,wh,1,B,B1645,645
3 | 2000-08-01 00:30:00 PDT,+0,+0,wh,1,B,B1645,645
4 | 2000-08-01 00:45:00 PDT,+0,+0,wh,1,B,B1645,645
5 | 2000-08-01 01:00:00 PDT,+0,+0,wh,1,B,B1645,645
6 | 2000-08-01 01:15:00 PDT,+0,+0,wh,1,B,B1645,645
7 | 2000-08-01 01:30:00 PDT,+0,+0,wh,1,B,B1645,645
8 | 2000-08-01 01:45:00 PDT,+0,+0,wh,1,B,B1645,645
9 | 2000-08-01 02:00:00 PDT,+0,+0,wh,1,B,B1645,645
10 | 2000-08-01 02:15:00 PDT,+0,+0,wh,1,B,B1645,645
11 | 2000-08-01 02:30:00 PDT,+0,+0,wh,1,B,B1645,645
12 | 2000-08-01 02:45:00 PDT,+0,+0,wh,1,B,B1645,645
13 | 2000-08-01 03:00:00 PDT,+0,+0,wh,1,B,B1645,645
14 | 2000-08-01 03:15:00 PDT,+0,+0,wh,1,B,B1645,645
15 | 2000-08-01 03:30:00 PDT,+0,+0,wh,1,B,B1645,645
16 | 2000-08-01 03:45:00 PDT,+0,+0,wh,1,B,B1645,645
17 | 2000-08-01 04:00:00 PDT,+0,+0,wh,1,B,B1645,645
18 | 2000-08-01 04:15:00 PDT,+0,+0,wh,1,B,B1645,645
19 | 2000-08-01 04:30:00 PDT,+0,+0,wh,1,B,B1645,645
20 | 2000-08-01 04:45:00 PDT,+9999,+4.52351,+4.52267,wh,1,B,B1645,645
21 | 2000-08-01 05:00:00 PDT,+0,+0,wh,1,B,B1645,645
22 | 2000-08-01 05:15:00 PDT,+0,+0,wh,1,B,B1645,645
23 | 2000-08-01 05:30:00 PDT,+0,+0,wh,1,B,B1645,645
24 | 2000-08-01 05:45:00 PDT,+0,+0,wh,1,B,B1645,645
25 | 2000-08-01 06:00:00 PDT,+0,+0,wh,1,B,B1645,645
26 | 2000-08-01 06:15:00 PDT,+0,+0,wh,1,B,B1645,645
27 | 2000-08-01 06:30:00 PDT,+0,+0,wh,1,B,B1645,645
28 | 2000-08-01 06:45:00 PDT,+0,+0,wh,1,B,B1645,645
29 | 2000-08-01 07:00:00 PDT,+0,+0,wh,1,B,B1645,645
30 | 2000-08-01 07:15:00 PDT,+0,+0,wh,1,B,B1645,645
31 | 2000-08-01 07:30:00 PDT,+0,+0,wh,1,B,B1645,645
```

Figure 35 The result occurred from our parser.

We created more than one parsers, because the format was not the same for all files. The parsers we created have a similar format with the file showed in the example above. Any differences seen are caused by the difference in formats between the input files.

3.2. Data processing and database formation

As we converted the data into a suitable form, we are ready to insert them into our database. To design and manage our database, we used Navicat. Navicat is a database administration tool that allows you to simultaneously connect to MySQL databases from a single application. [18]

So, after we parsed our data, we inserted them to our database and we organized them into tables for better management. After the insertion of our data, our database is comprised of the following tables:

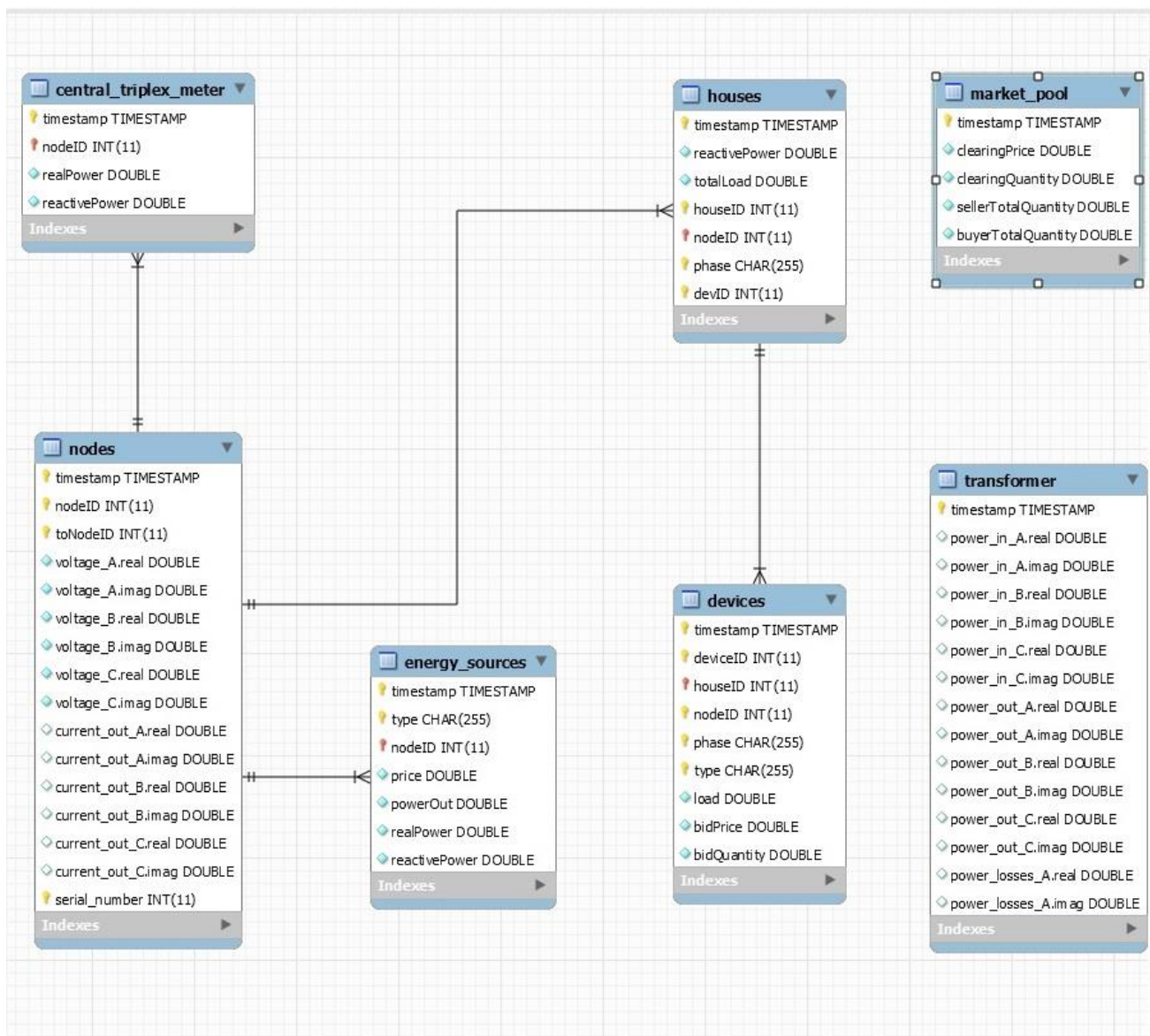


Figure 36 Our Database Formation.

Nodes In this table we will find information about each node. Each node has a unique node id. This table contains the voltage (real and imaginary) per phase for each node, as well the current (real and imaginary) per phase which flows from one node to another. It contains also the node id towards which the current is flowing.

Central_triplex_meter This table contains all information belonging to central triplex meter. It contains the identification number of each node included to the triplex meter. Each node has a real power and a reactive power which are inherent to the node they belong to.

Houses In this table we will find information for each house. Each house has a unique house id and it belongs to a specific node. Each house can have one or more power phases, but only one at any given time. Also this table contains the total load consumed by a house and its reactive power at any given time.

Devices This table contains information about the devices belonging to a house. Each device has a unique device id, which is closely related to the house to which it belongs. It also contains the phase and the type of the device, as well as the bid price and the bid quantity of each device and the total load it actually consumed.

Energy_sources In this table we will find information about energy sources of the system. This information includes the node in which the energy source belongs, the type of energy source (e.g. Diesel or Wind), its real and reactive power and the price it sells energy.

Market_pool This table contains all information related to market. A market contains aggregated data for clearing price, clearing quantity, buyer's quantity and seller's quantity.

Transformer In this table we have information only about the transformer of the system. The data related to the transformer are only the power (real and imaginary) per phase.

Finally, all the tables have a field of timestamp, to observe the changes that occur every single time.

The data change every 15 minutes and the experiment conducted for 7 days (one week).

Chapter 4

4. System Implementation

4.1 Architecture

In this chapter we will describe the architecture of our web application. A Web Application is defined as a software which is accessible through the World Wide Web (Internet). For this purpose there are two main parts that allow users to interact with the application:

1. A web server which serves various user requests and returns as a response HTML files.
2. A Database with which the web server communicates to store, retrieve and modify data necessary for the operation of the application.

The major advantage of internet applications is that on the user's part (client side) is not required any software other than a browser, like Mozilla Firefox, Google Chrome, Safari, Opera or Internet Explorer.

In our web application we use Node.js as our server, a cross-platform runtime environment, which contains a built-in library to allow applications to act as a web server without software such as Apache HTTP Server or Nginx. We will be more specific about Node.js and its server-side web frameworks, such as Express.js and Sails.js in the following subsections. As database we chose to use MySQL, which is a Relational Database Management System (RDBMS), it is simple to install and suitable for the database model we created, both in terms of management functions and the relationships between the tables. Thus, users using a browser communicate with the Node.js Server of our web-application and so they have access to our application. To transfer data between client and the server the HTTP protocol is used, also known as the "Internet Protocol". The HTTP protocol is used to load webpages at the request of a user - client. Because our implementation supports asynchronous communication (without loading or updated the website) data transfer can be done using the AJAX technology. [1]

4.2. Used Technologies

To implement a web application it is essential to adopt specific techniques and to integrate several technologies that help web developers depending on the result they want to achieve. The two most important elements for implementing a web application is the server-side programming language to be used as well as the client-side language programming for processing the dynamic data displayed on the website.

To build this web-application, we used a variety of web technologies and tools. For the server-side programming language we used JavaScript. Nowadays JavaScript is widespread both as client and as server programming language. In client-side, we also used JavaScript with suitable frameworks. The main tool that we used in the frontend part is the AngularJS Framework, while in the backend we primarily used the Node.js platform in conjunction with the Sails.js Framework.

4.2.1 Backend Technologies

In this paragraph we will refer the technologies we used for development of our system in backend.

4.2.1.1. Node.js

Node.js is an open source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System Z and IBM i. It provides an event-driven architecture and a non-blocking I/O API designed to optimize an application's throughput and scalability for real-time web applications. It uses Google V8 JavaScript engine to execute code, and a large percentage of the basic modules are written in JavaScript. Node.js contains a built-in library to allow applications to act as a web server without software such as Apache HTTP Server, Nginx or IIS. [1, 19]

4.2.1.2. Express.js

Express.js is a Node.js web application server framework, designed for building single-page, multi-page, and hybrid web applications. It is the de facto standard server

framework for Node.js. It is relatively minimal and flexible and provides a robust set of features for web and mobile applications. [1, 20]

4.2.1.3. Sails.js

Sails.js is a web framework (placed over Express.js framework) that makes it easy to build custom, enterprise-grade Node.js apps. It is designed to resemble the MVC architecture, but with support for the more modern, data-oriented style of web app development. It's especially good for building realtime features. [21]

4.2.1.4. MySQL

MySQL is an open source RDBMS (Relational Database Management System) that uses SQL (Structured Query Language). SQL is the most popular language for adding, accessing and managing content in a database. It is most noted for its quick processing, proven reliability, ease and flexibility of use. [1, 22]

4.2.1.5. Npm

Npm is a package manager for JavaScript, and is the default for Node.js. Npm is a command-line tool for interacting with a huge repository of Node.js projects. It can install packages in local or global mode. [1, 23]

4.2.2. Frontend Technologies

In this paragraph we will refer the technologies we used for development of our system in frontend.

4.2.2.1. HTML / HTML 5

HTML (Hyper Text Markup Language) is the standard markup language used for structuring and presenting content on the World Wide Web. HTML5 is the latest evolution of the standard that defines HTML. [1]

4.2.2.2. CSS / CSS 3

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. Although most often used to change the style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications. CSS 3 is the latest evolution of the standard that defines CSS and is divided into several separate documents called "modules". [1]

4.2.2.3. JavaScript

JavaScript is a high level, dynamic, untyped and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three essential technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern web browsers without plug-ins. JavaScript is also being used in server-side programming (such as Node.js), game development and the creation of desktop and mobile applications. [1]

4.2.2.4. AngularJS

AngularJS is an open-source web application framework. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model-view-controller (MVC) and model-view-view model (MVVM) architectures, along with components commonly used in rich Internet applications. The AngularJS library works by first reading the HTML page, which has embedded into it additional custom tag attributes. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources. [1]

4.2.2.5. Twitter Bootstrap

Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML- and CSS-based design templates for typography, forms,

buttons, navigation and other interface components, as well as optional JavaScript extensions. It aims to ease the development of dynamic websites and web applications. Bootstrap is a front end framework, that is, an interface for the user. [1]

4.2.2.6. jQuery

jQuery 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. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript. [24]

4.2.2.7. Ajax

Ajax (short for asynchronous JavaScript and XML) is a group of interrelated Web development techniques used on the client-side to create asynchronous Web applications. With Ajax, web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Ajax is not a single technology, but a group of technologies. HTML and CSS can be used in combination to mark up and style information. The DOM is accessed with JavaScript to dynamically display – and allow the user to interact with – the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads. [1]

4.2.2.8. JSON

JSON (short for JavaScript Object Notation), is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is the primary data format used for asynchronous browser/server communication (AJAJ), largely replacing XML (used by AJAX). [1]

4.3. MVC Architecture

However, developer challenges do not end with the choosing of suitable web technologies. One of the major challenges we had to face was the selection of a proper architecture. The web architecture which forms the basis of developing our web application is the MVC Architecture.

MVC (short for Model–View–Controller) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. Let's take a look a little deeper:

Model

Model is connected with our application's database. It stores data that is retrieved according to commands from the controller and displayed in the view. The model doesn't know anything about views and controllers. The Model part in our web application is everything related to our database. In the Model belong all sails.js files, which are the connection between our application and our database. To be more specific each js file in the server side of our application is a query from our application to database, and the result (answer) of that query is used to represent to view.

View

View is what's presented to the users and how users interact with the app. The view is made with HTML, CSS, JavaScript and often templates, in our web application we use the AngularJS template. View can easy integrate with the Model through Controller via AJAX technology (HTTP request). Through controller we define the client side template that will represent the results that come from a specific file of our server. Angular templates are the View in our web application.

Controller

The controller is the decision maker and the glue between the model and view. The controller updates the view when the model changes. It also adds event listeners to the view and updates the model when the user manipulates the view. The controller contains all the business-logic of our application. It decides which server side file (model) corresponds to each client side template (view). [1, 25, 26]

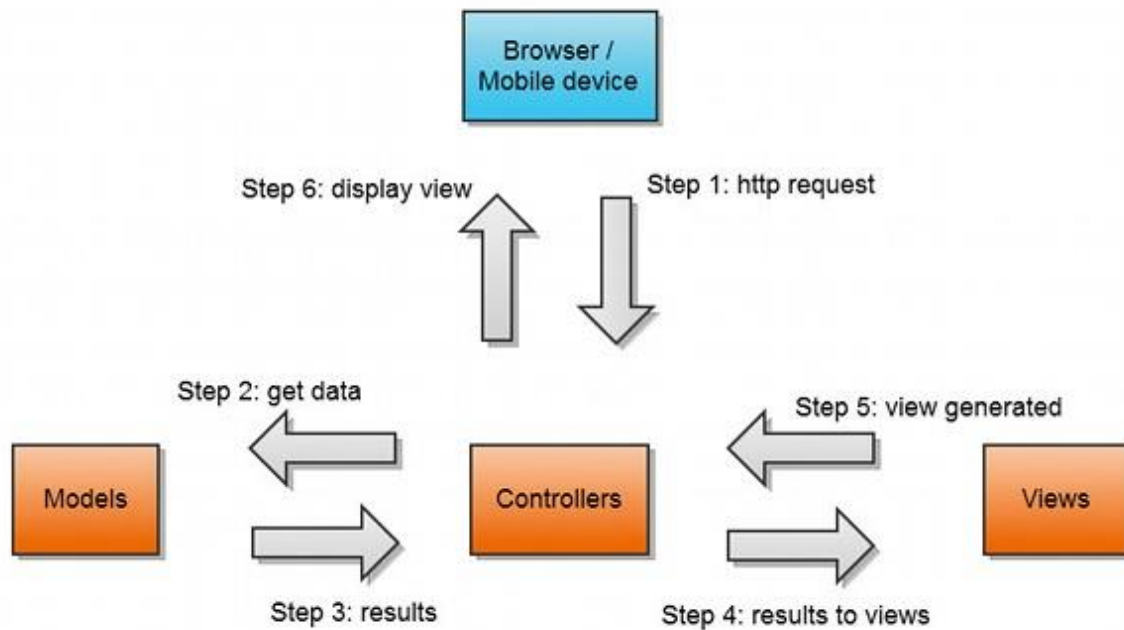


Figure 37 Schematic illustration of MVC Architecture.

4.4. User Interface

In this subsection we will deal with the implementation of our web application. As we already mentioned, GridLab-DVisor uses data derived from experiments conducted based on GridLAB-D simulation platform. Our goal was to succeed in presenting raw data coming from GridLAB-D using proper visualization techniques, so that patterns can be easily and quickly spotted by the user. In the next paragraphs we will analyze the way in which we developed our application. We will describe the interface used by the user to navigate our application. To support the functions a user can perform, we developed an easy to use and understandable interface. On the main page (index page)

general information regarding our web application is displayed. Such information includes all the tables that are available in our implementation. Each table has an accompanying description and a picture showing a highlight of the table. On top of the index page, the user can also see a navigation bar, which has some extra information about the application, such as information on the used technologies and the curriculum vitae of the developer. In the navigation bar also exists a dropdown menu from which the user can select the desired table.

GridLabDVisor Home Used Technologies About Developer Select Table

Welcome to GridLabDVisor!

GridLabDVisor is a web application which collects data from a power distribution system simulation and analysis tool (gridLab-D) and uses javascript libraries to represent these data in forms and diagrams which are useful and understandable for users.

[Learn more](#)

To start the navigation in the application choose the table that you are interested in, from the following list. You can also find the tables from the bar to the top of the page, by clicking at "Select Table" selection and choosing the table that you are interested in.

Central Triplex Meter

This table contains information about Central Triplex Meter of the System. To be more specific contains the values for Real & Reactive Power for each central node of the system per day.

[Go to graph](#)

Devices

This table contains information about all Devices located in a house. Each device has a Bid Price and a Bid Quantity that beats in the auction and the Consumed Load that actually takes.

[Go to graph](#)

Energy Sources

This table contains information about Energy Sources of the System, which are divided to three categories. It can inform you about Real & Reactive Power for each category per node, as well as the Price that the Energy is selling.

[Go to graph](#)

Houses

In this table you can check the actual Load that inserts into your house and the percentage of this that become Reactive Power and not used as power of house anymore.

[Go to graph](#)

Market Pool

This table contains information about the Market Pool of the System. So watching this table you can check how is formed the Market Pool of System per day.

[Go to graph](#)

Nodes

In this table you can check how the Voltage and the Current is flowing among the Nodes of the System and how the System actually is formed for each day per phase.

[Go to graph](#)

Transformer

In this table you can check the value of the Power that gets in the Transformer of the System, as well as the Power that gets out and the Losses per phase.

[Go to graph](#)

Devices Total Table

This is a table that contains all the information about Devices in table format. Contains exactly the same information with "Devices" table, but with different format.

[Go to graph](#)

System Graph

In this table you can watch the topology of our system, ie that you are able to see how the nodes of the system are connected, as well as how many houses are connected to each node.

[Go to graph](#)

GridLAB-D

GridLAB-D™ is a new power distribution system simulation and analysis tool that provides valuable information to users who design and operate distribution systems, and to utilities that wish to take advantage of the latest energy technologies.

[View details >>](#)

Vis.js

Vis.js is a dynamic, browser based visualization library. The library is designed to handle large amounts of dynamic data, and to enable manipulation of and interaction with the data.

[View details >>](#)

Highcharts

Highcharts is a charting library written in pure JavaScript, offering an easy way of adding interactive charts to your web site or web application. It works in all modern mobile and desktop browsers.

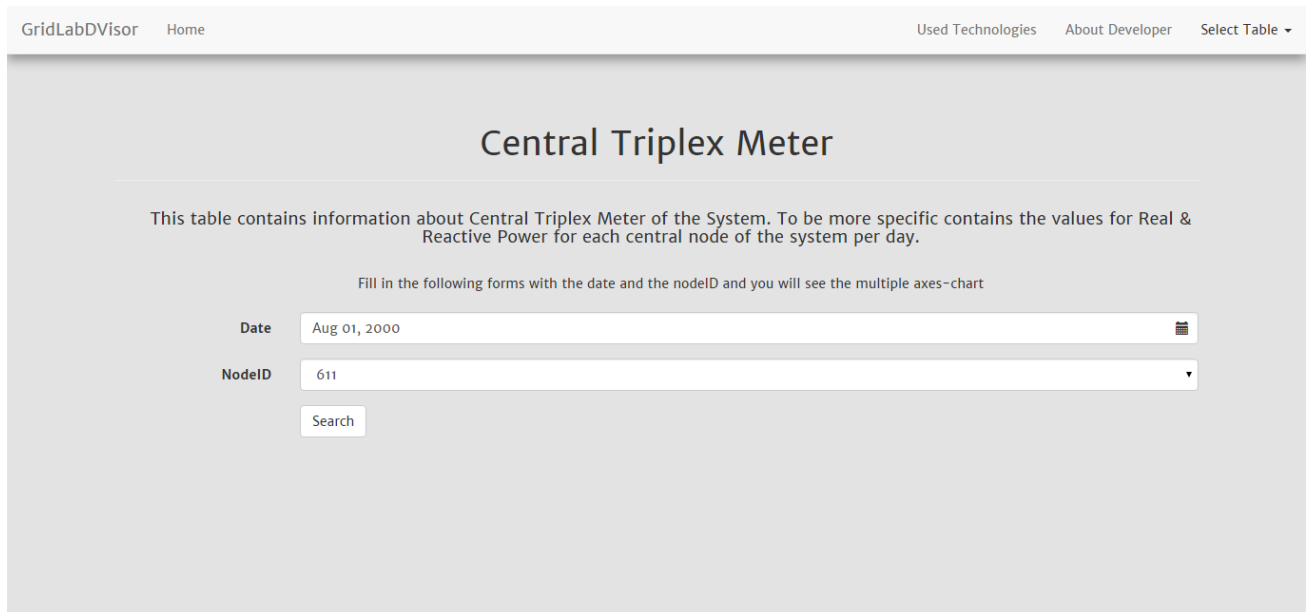
[View details >>](#)

Figure 38 The index page of GridLab-DVisor.

Each table of our web based application contains unique data, which it graphically presents. In the following paragraphs we are going to analyze these tables.

Central Triplex Meter

Let's start with the "Central Triplex Meter" table. When we select this table and before the graph is presented, appears a web page that asks from the user some extra information about what they want to see. Specifically, this page requests the Date and the identification number of the node that users would like to see on the graph, as shown below.



The screenshot shows a web application interface for the "Central Triplex Meter". At the top, there is a navigation bar with "GridLabDVisor" and "Home" on the left, and "Used Technologies", "About Developer", and "Select Table" on the right. The main content area has a title "Central Triplex Meter" and a description: "This table contains information about Central Triplex Meter of the System. To be more specific contains the values for Real & Reactive Power for each central node of the system per day." Below this, there is a prompt: "Fill in the following forms with the date and the nodeID and you will see the multiple axes-chart". The form consists of two input fields: "Date" with the value "Aug 01, 2000" and a calendar icon, and "NodeID" with the value "611" and a dropdown arrow. A "Search" button is located below the NodeID field.

Figure 39 The selection form used for Central Triplex Meter table.

When users click on the "Date" input, appears a pop-up window, which contains a calendar from which they can choose the date they prefer. We created this calendar, in such a way that prevents the selection of invalid dates. Valid date is considered a date for which we have corresponding data in our database. In our web application valid are only the dates between 01 August 2000 and 07 August 2000, as shown in Figure 39.

When we click on the "Node ID" input, a list of all nodes of the system is displayed to the user. The user can select only one of the displayed nodes.

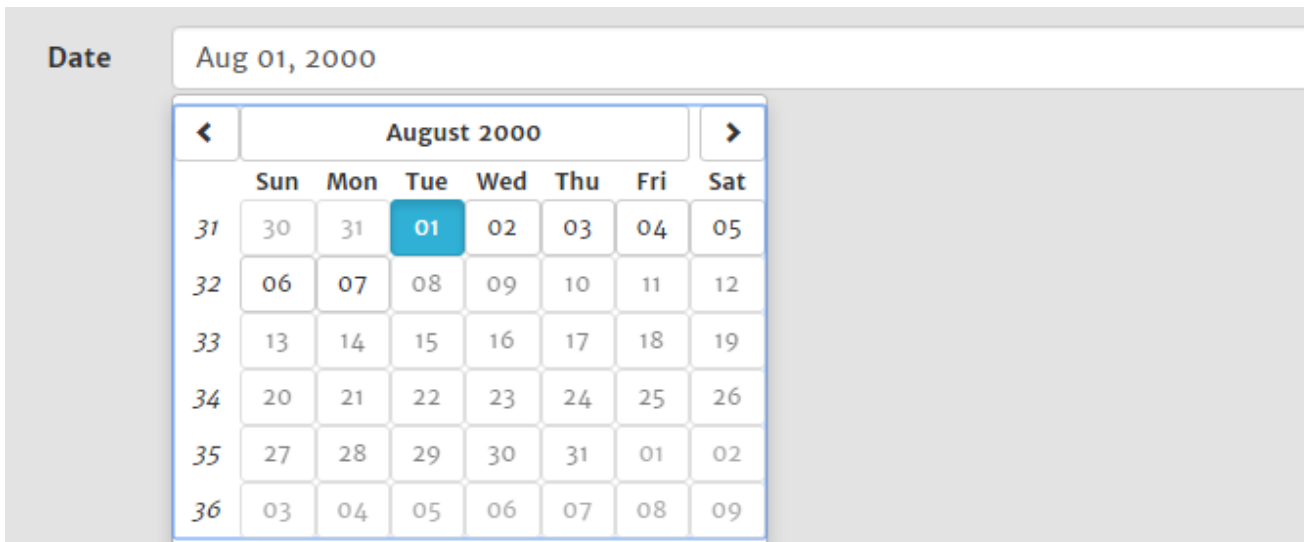


Figure 40 The calendar displayed when the user press the Date form.

So, when a user fill in the forms we mentioned, and presses the “Search” button a graph of the Central Triplex Meter is going to be displayed.

The information presented in Central Triplex Meter is the Real and the Reactive Power of the node we have chosen. In Figure 40, we show the Central Triplex Meter’s graph in a “Multiple Axes” Diagram. A multiple axes diagram, has a separate y-axis for each variable showed. We used a multiple axes diagram, since it is very accurate because of the number of its axes.

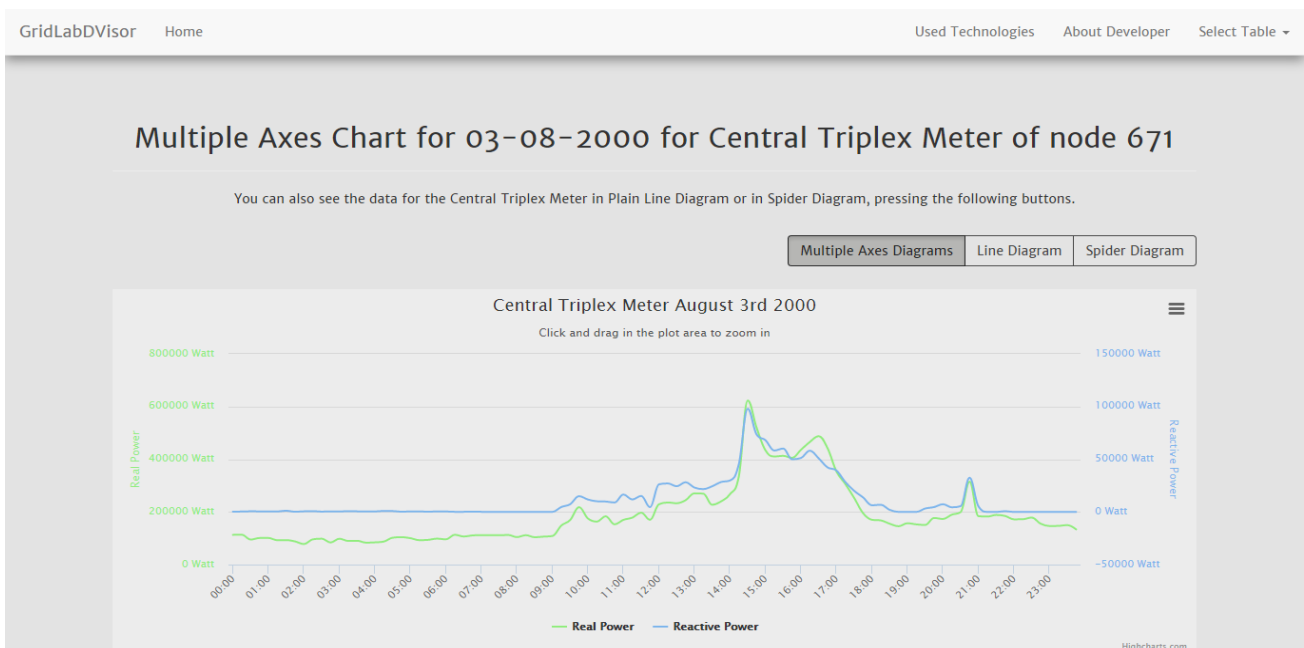


Figure 41 Multiple Axes Chart for Central Triplex Meter.

Besides “Multiple Axes Diagram”, we decided to insert line and spider diagrams for this table. Although these charts are simple, they easily present the pattern of the data. Watching these diagrams, the user is able to understand easily and quickly the information received.

In the Line Diagram we just present the data for Real and Reactive power in lines. Since data are continuous, they can be easily presented as line diagrams producing a clear and appealing result. One of the main differences between the Line Diagram and the Multiple Axes Diagram is that the Line Diagram helps the user distinguish any patterns appearing in each graph. Also, since the two graphs are rendering inside the same canvas it is easier for the user to understand the correlation between them. This helps the user understand more quickly and easily the behavior of the system.

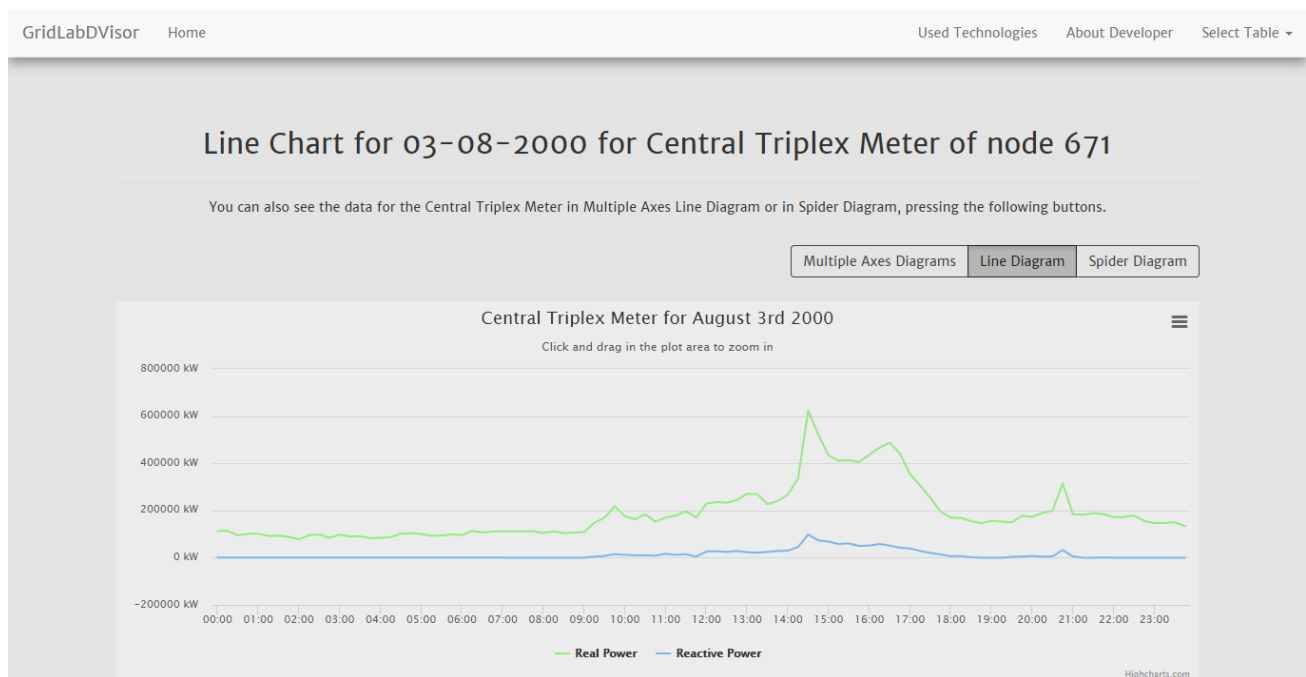


Figure 42 Line Diagram for Central Triplex Meter.

Another diagram we chose to present data of the Central Triplex Meter table is the Spider Diagram. We selected the Spider Diagram for the same reason we chose the Line Diagram; to give the user a better view of system behavior. Watching the diagram produced based on the Central Triplex Meter data (Figure 42), we observe that it is indeed very easy to discern patterns in data.

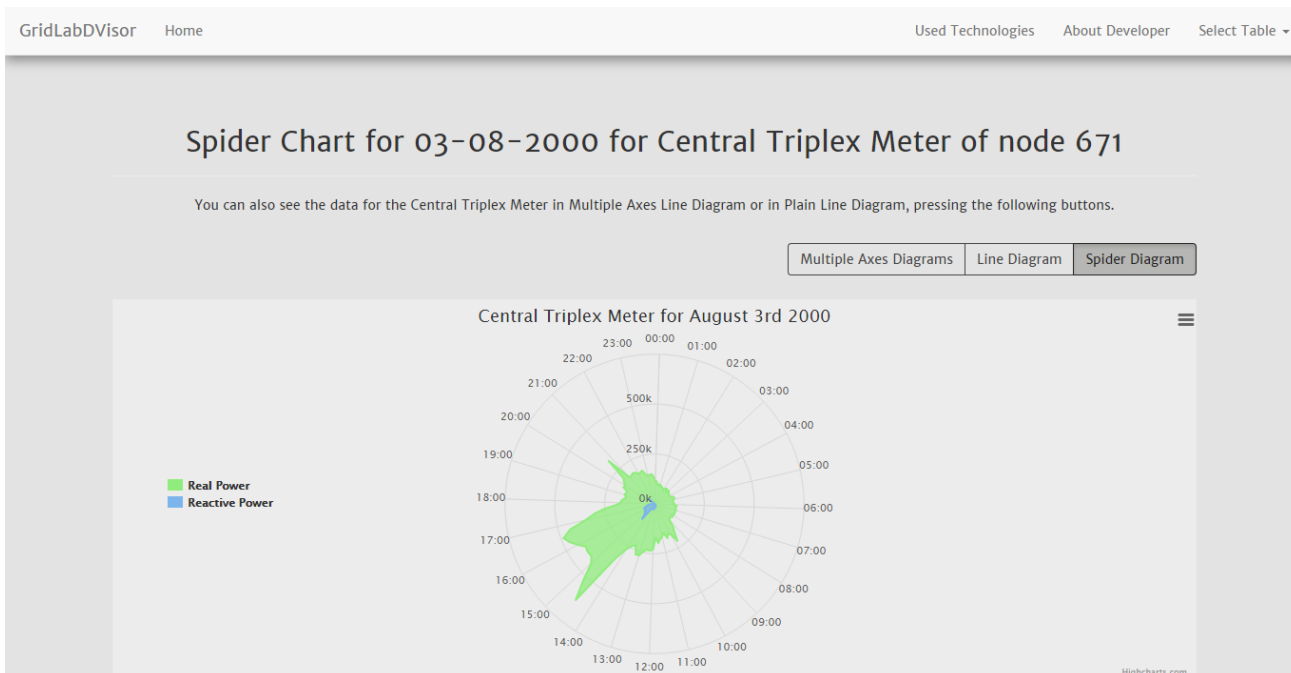


Figure 43 Spider Diagram for Central Triplex Meter.

Devices

The next table on our list is the Devices table. In this table the user can select one device of any house and view its behavior during the date period they have chosen. The following picture shows the forms used for the device selection.

Figure 44 The selection form used for Devices table.

To present the diagram for the Devices table, the user has to fill in some forms first. When the user clicks on the “Date” input, appears the calendar we analyzed above. Subsequently, when a user clicks on the “Type” input, a list of all devices belonging to the house will show up. In this web application the only devices belonging to a house are the Air Condition and the Water Heater. The next input the user encounters is the “Node ID” input. As we already mentioned, this form displays a list of all the nodes of the system, when a user clicks on it. Finally, and after the user has select the identification number of the node, they have to select the id of the house in the “House ID” input. The list of houses loads after the user has chosen the node, because each node has different houses, for example House ID 14645 belongs to node 645 and should not be shown as part of any other node. After completing all the forms, the user can press the “Search” button to display the diagram for the selected device.

Each device has a Bid Price and a Bid Quantity for the auction and the Consumed Load that actually consumes. In the charts we have created, we are showing these factors all together and individually. We are using here too, the multiple axes diagram, because of its accuracy.

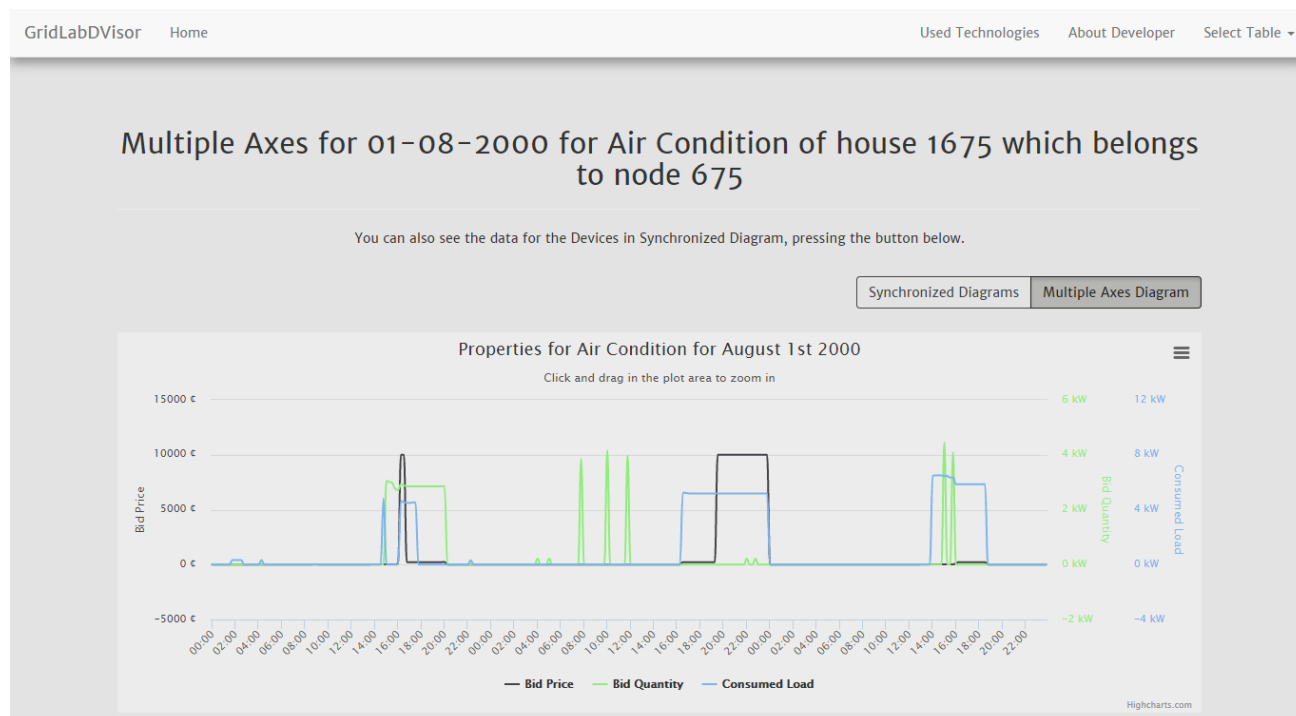


Figure 45 Multiple Axes Diagram for Devices table.

Another diagram we used in this table is the “Synchronized Diagrams”. In this diagram each factor presents individually from the others. Actually they are three different diagrams one below the other, where all they have in common is the x-axis value when the mouse hovers one of them. To be more specific when the mouse hovers one of these sub-graphs, takes the value of the x axis of the sub-graph, and sends it in the other two sub-graphs. Thus, all the graphs show their values for a specific value of x-axis, but is separately rendered, so as to have the maximum possible accuracy.

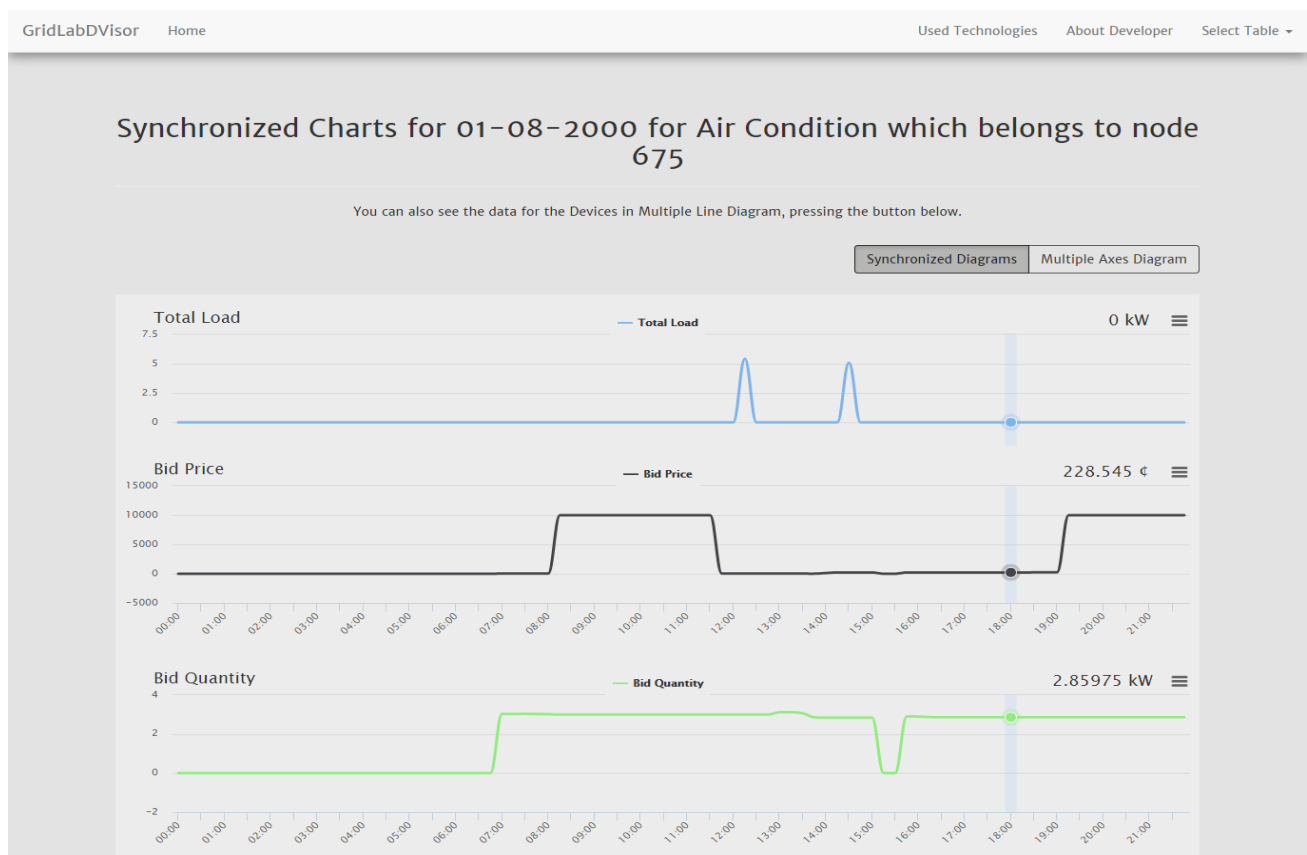


Figure 46 Synchronized Diagrams for Devices table.

For the devices of a house, we also inserted another table, named “Devices Table Total”. This table includes exactly the same information with the “Devices” table, but in different form. This table is in the heatmap form. In the x-axis it has all days of the week, and in the y-axis it has all the hours of each day. As we already mentioned the data were given to us, was for 7 days and the data was changed every 15 minutes. So, for this format of the table, we had to create a table which in the x-axis was containing

seven dates (01 – 07 August 2000) and in its y-axis was containing daylight hours per quarter (00:00 – 23:45).

We developed this heatmap form for each factor of this table, i.e. for Bid Price, Bid Quantity and the Total Load. The following picture shows the Consumed Load in heatmap form. The same format have and the other two characteristics of the devices.

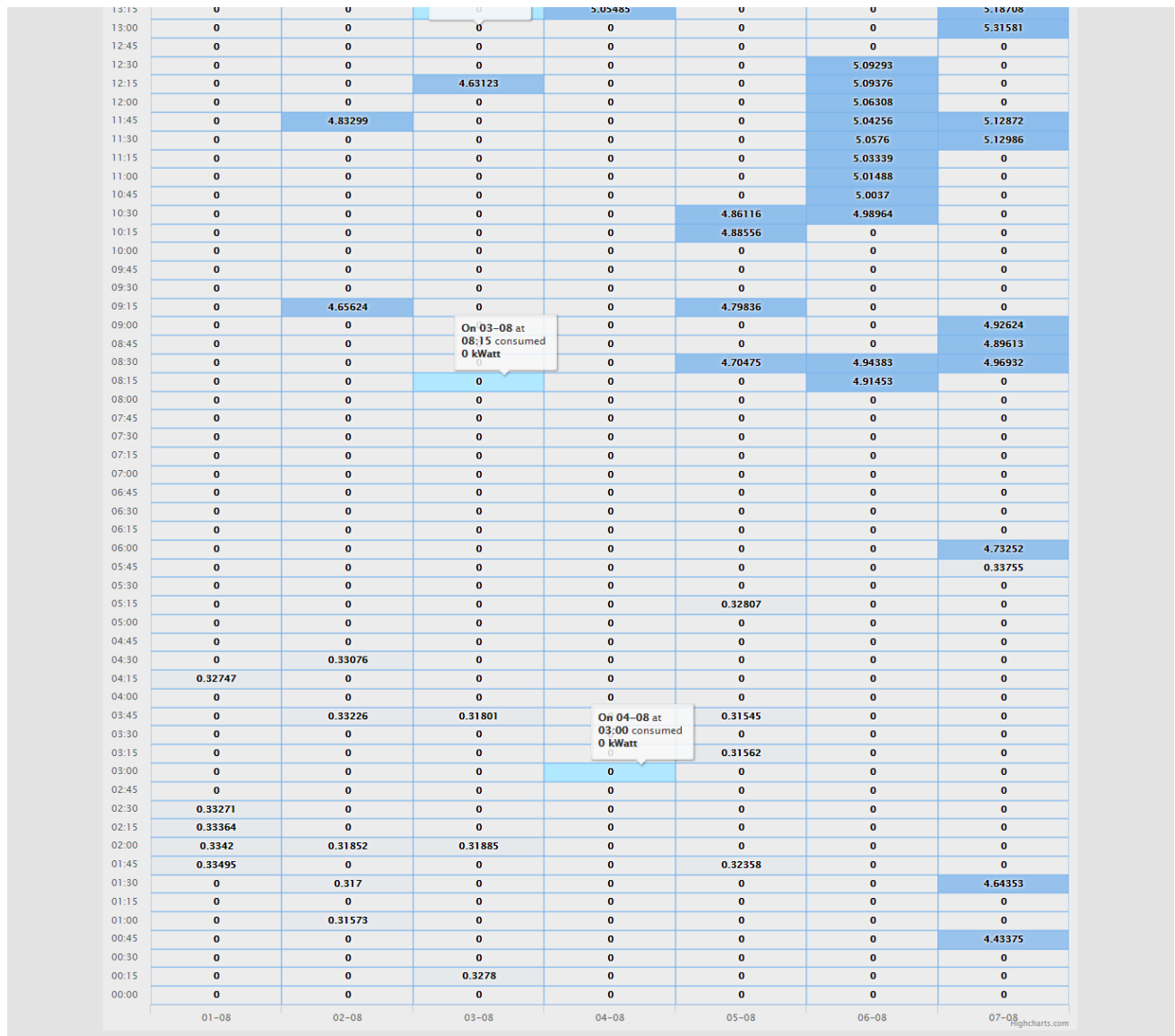


Figure 47 The Consumed Load in Heatmap form of the Devices Total Table.

Energy Sources

Another table of our web application is the “Energy Sources”. In this table users have to select the type of energy source (Diesel, Wind or Solar) that they would like to review for a specific node and date. The selection form we created for this table is similar to the selection pages we already analyzed. This table includes information about the Real and the Reactive power for each category of energy sources per node, as well as the price that the energy is selling. The chart types we chose for this table are the multiple axes chart, the line and the spider diagram. Because the values of each factor have very little differences between them, only the multiple axes diagram gives the required accuracy, as we can see in the following figure.

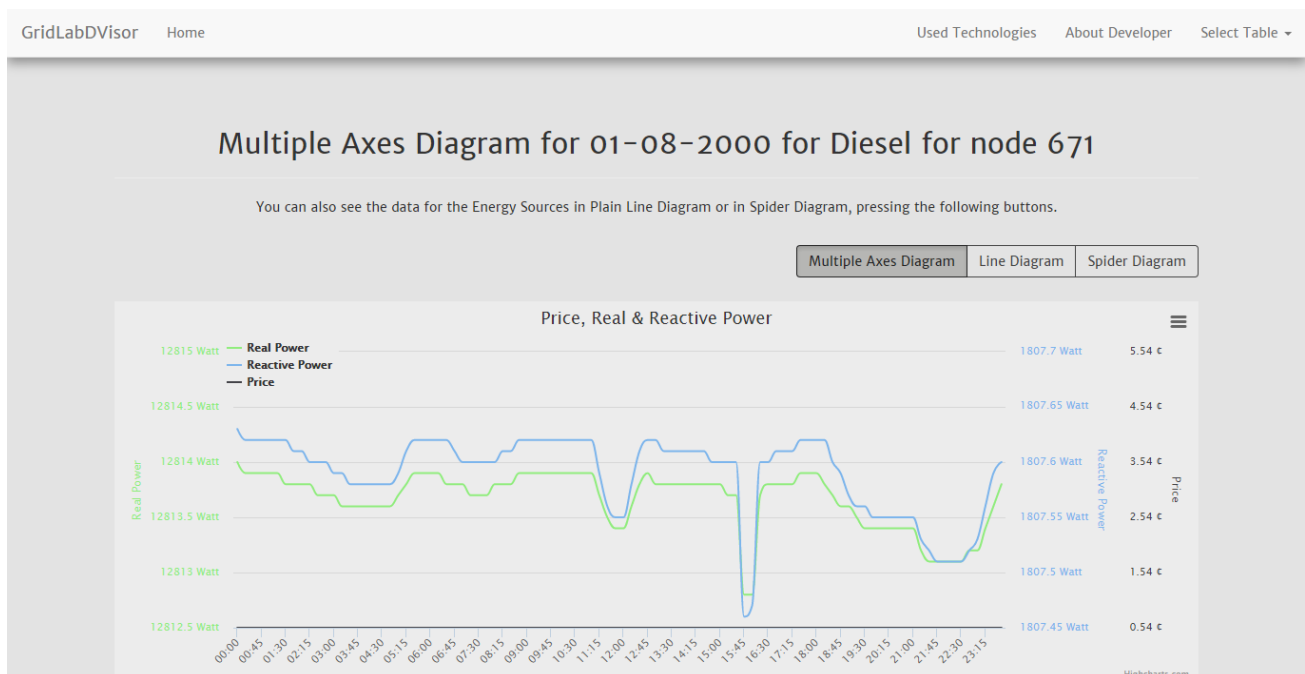


Figure 48 Multiple Axes Diagram for Energy Sources Table.

The other two diagrams (the Line and the Spider Diagram) do not have the same accuracy as the first one, due to the different orders of magnitude, but we used them exactly for this reason. They show the behavior of the system; to be more specific, from the Line and the Spider Diagram we can understand that the difference of two sizes (Real and Reactive power) is big, while the Multiple Axes Diagram cannot give us such an overview of the system. The Line and the Spider diagram follow.

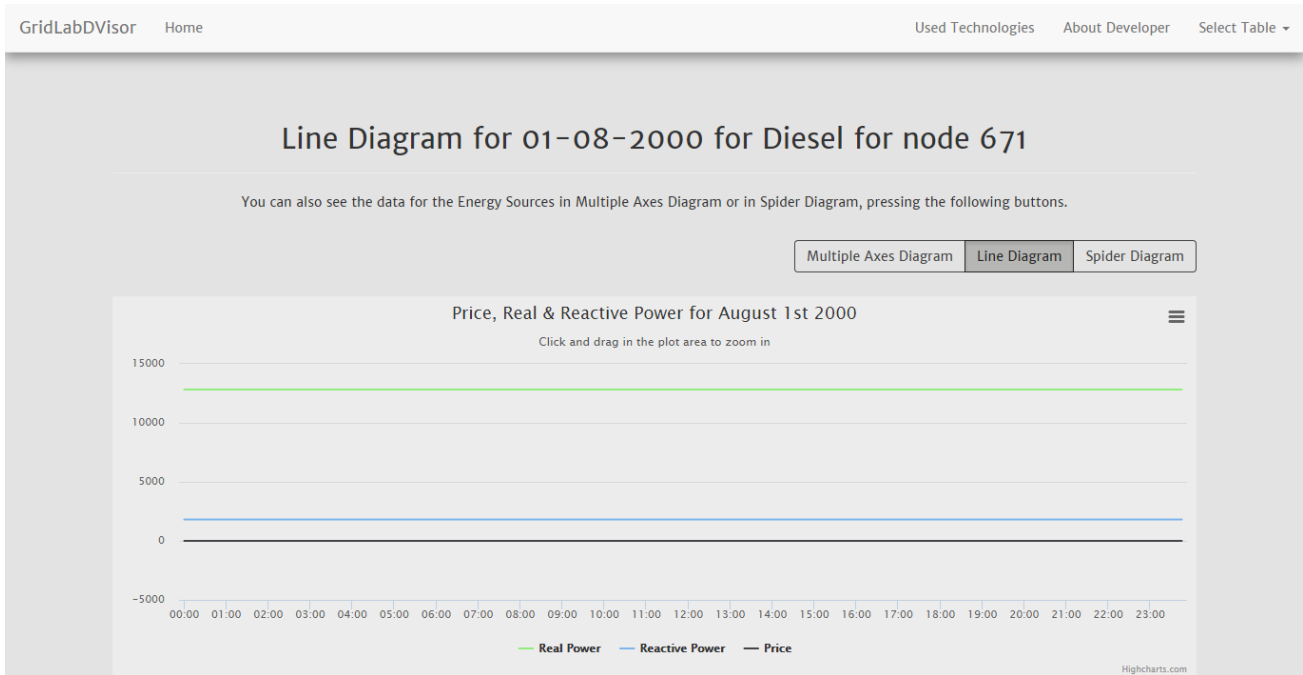


Figure 49 The Line Diagram for Energy Sources table.

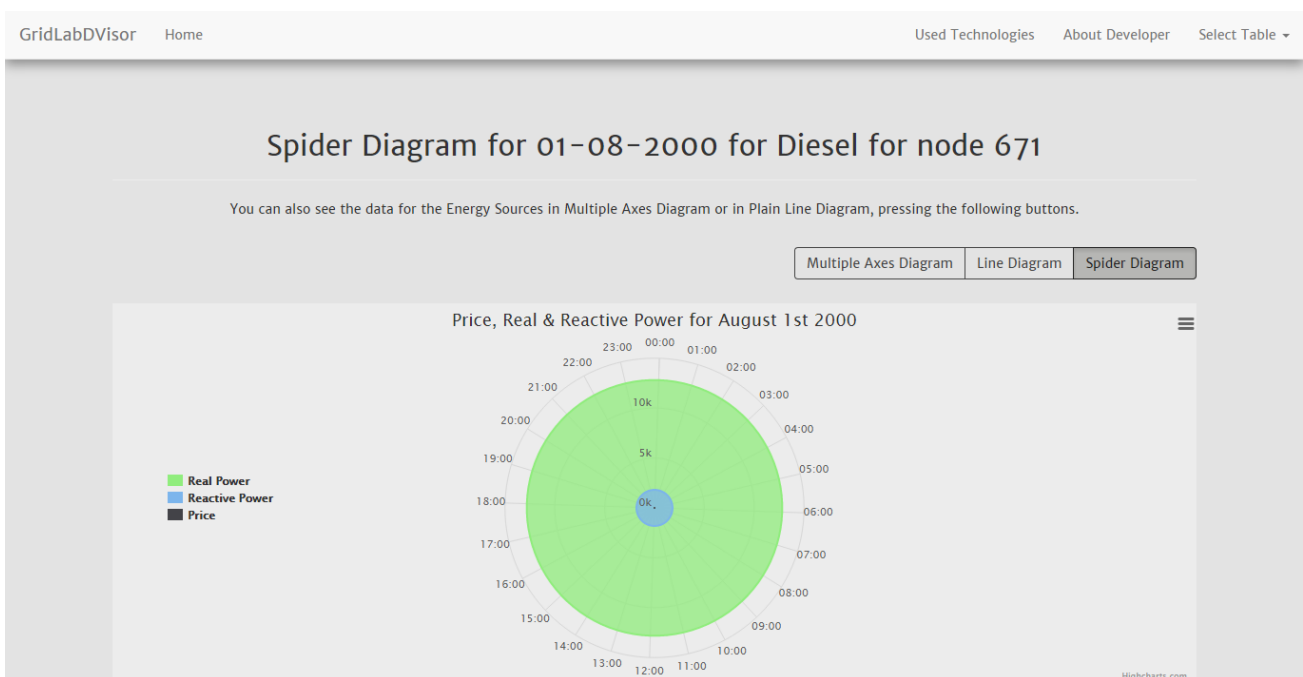


Figure 50 The Spider Diagram for Energy Sources table.

Houses

Another very important table that we have in our web application is the “Houses” table. In this table the user can view the total load each house of the system consumed and how much of this load turns into reactive power. Because each house may belong to more than one phase (e.g. it probably belongs to phases A and B), we had to take that into consideration. For this reason, in the select form we created for this table it was necessary to use jQuery to manage DOM manipulation. The problem we faced on this page was that multiple phases probably exist in each house. And that’s why we were “forced” to use DOM manipulation. In this select page, once users have selected a date, they need to select the id of the node the house belongs to. When the user selects the node id, an HTTP request is sent to the server, which is then translated to a query to our database and finally returns the phases belonging to the particular node. Then, after the user selects one of the available phases, in the same manner, a second query to our database is sent, which returns all houses belonging to that phase of the selected node. This select page we developed is very similar to the other pages we described earlier, as you can see below.

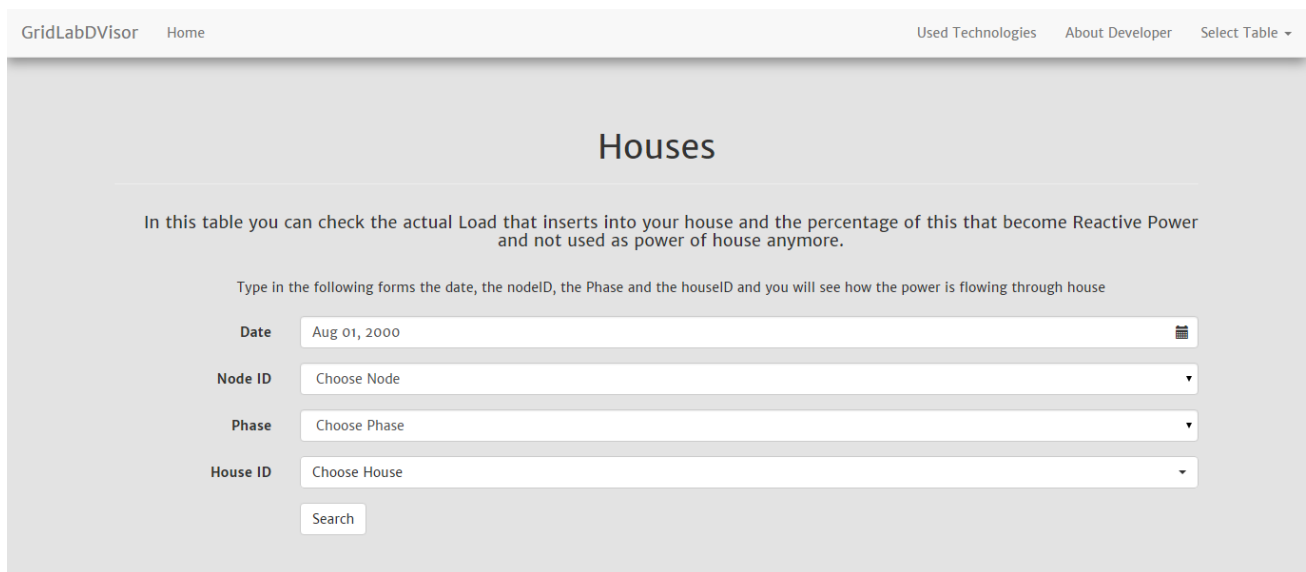


Figure 51 The selection page for the Houses table.

As you can see, this selection form contains four inputs, in which the user can choose the date, the id of the node the house belongs to, the phase of the house and finally, the identification number of the house. The “Date” input is a calendar, as we already analyzed in previous paragraphs. The “Node ID” input is a list containing all nodes of

our system. The “Phases” input is a list of all phases that are available to each node, and finally, the “House ID” input is a list containing all houses belonging to a specific phase of the node we selected in the second input. Here is an example of how we can select one or more houses of node 671. At first, we selected the day for which we would like to view the chart. The next step was the selection of a node, so we chose node 671. By clicking on the next form (“Phases” form), a list of the available phases shows up. Node 671 contains all three available phases: A, B and C.

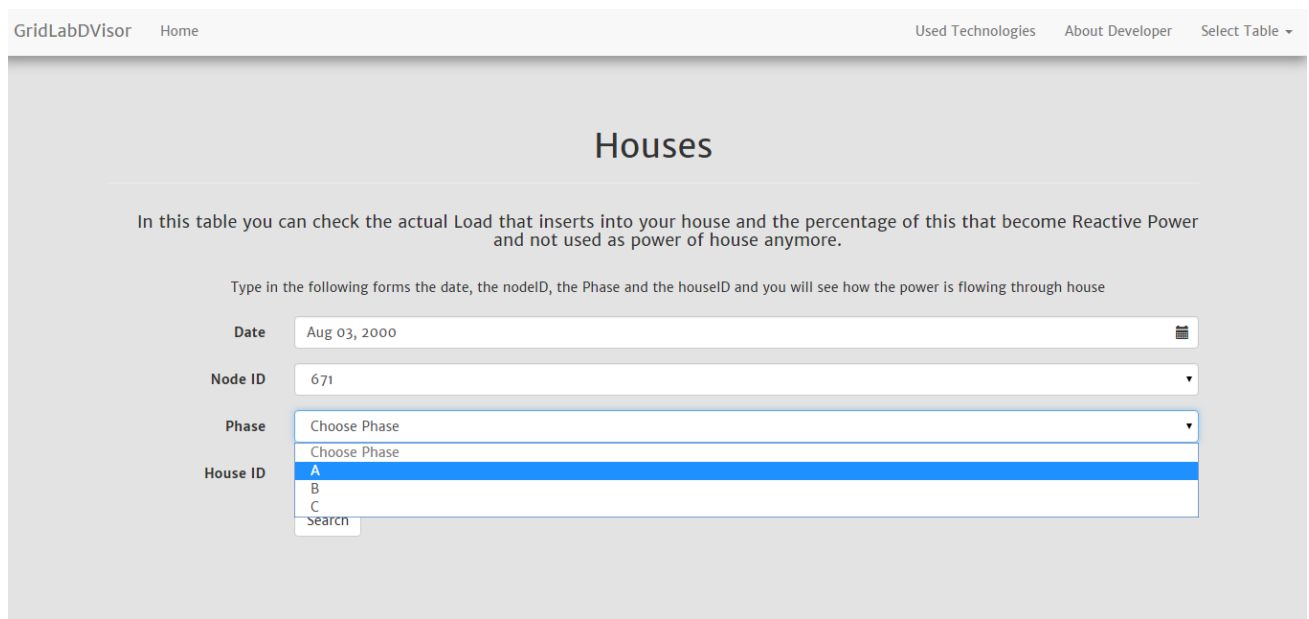


Figure 52 The selection of phase in the Houses table.

In this example, we chose phase A of node 671. Then, once we have completed the forms above, the list of houses belonging to the categories we have selected becomes available. So, by clicking on the “House ID” input, a pop up window containing the list of the houses shows up.

From this list, the user is able to select one or more houses, to review their power consumption. We thought that it could be useful for the user to have the ability to compare the consumption of their house to the consumption of other houses. In that way the user can better analyze some aspects of the consumption of their house that was not able to distinguish so far.

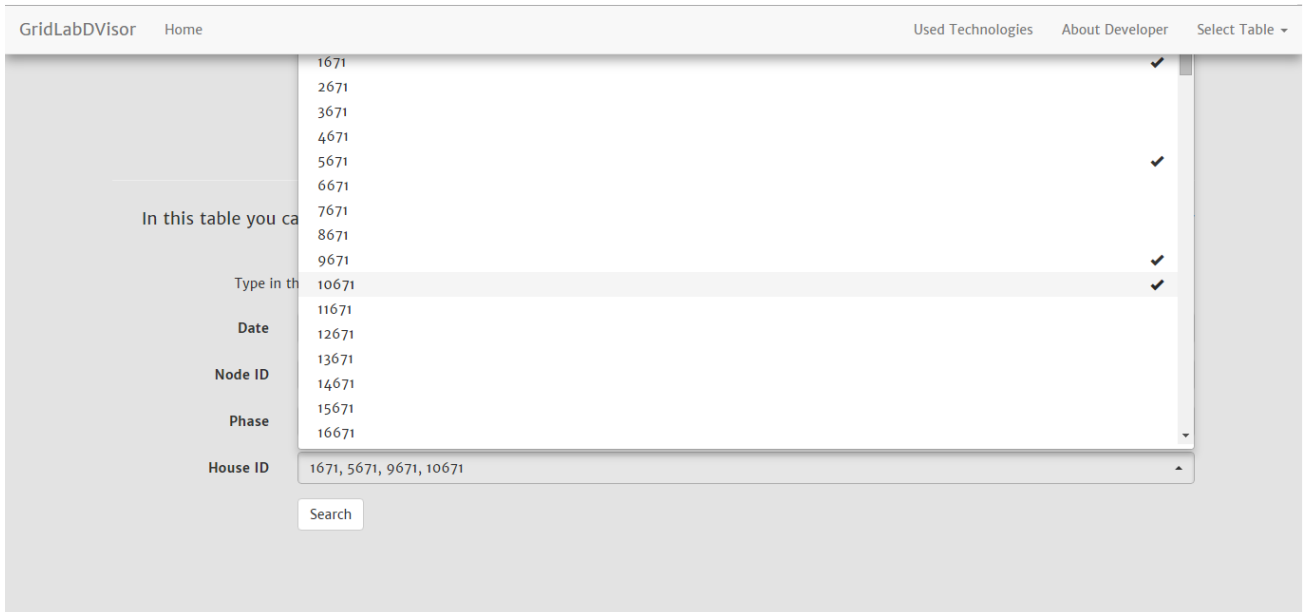


Figure 53 The selection of houses in the Houses table.

Each house's energy consumption is presented either as a Line or a Spider Diagram. Both charts are simple, but it is very easy for the user to distinguish the patterns of the data. We selected these diagrams, because we thought the users could easily and quickly get a general idea of their house's energy consumption.

In the following figures (Figure 53 and Figure 54), you can see the results for more than one house in both the line and the spider chart.

Line Chart for 03-08-2000 for node 671

You can also see the data for House in Spider Diagram, pressing the button below.

Line Diagram Spider Diagram

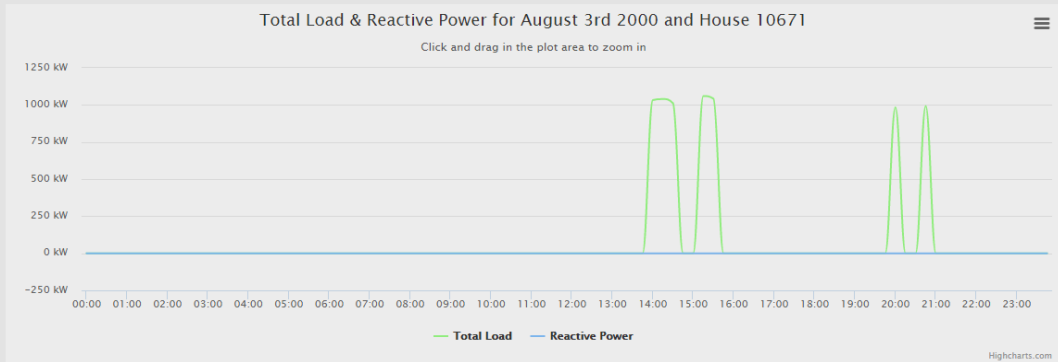
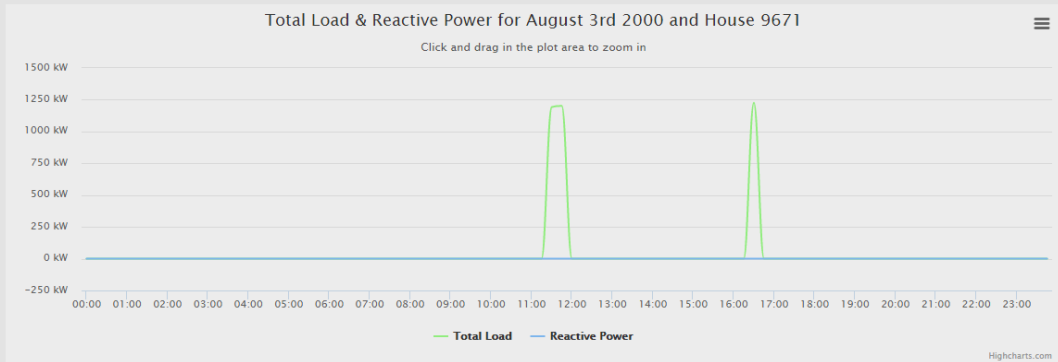
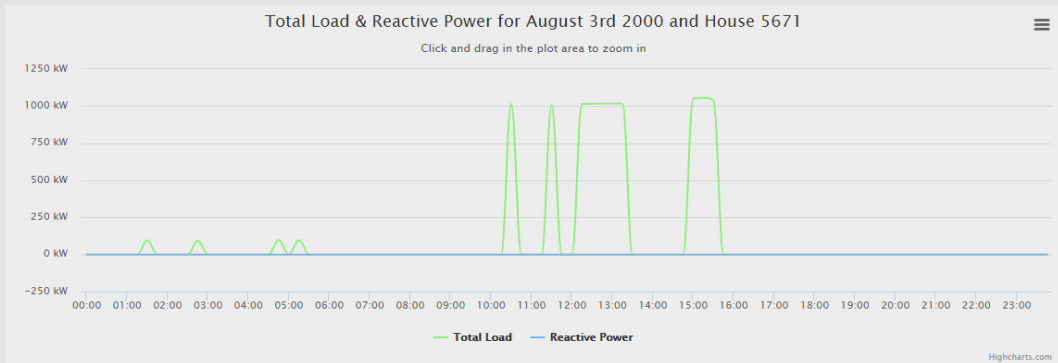
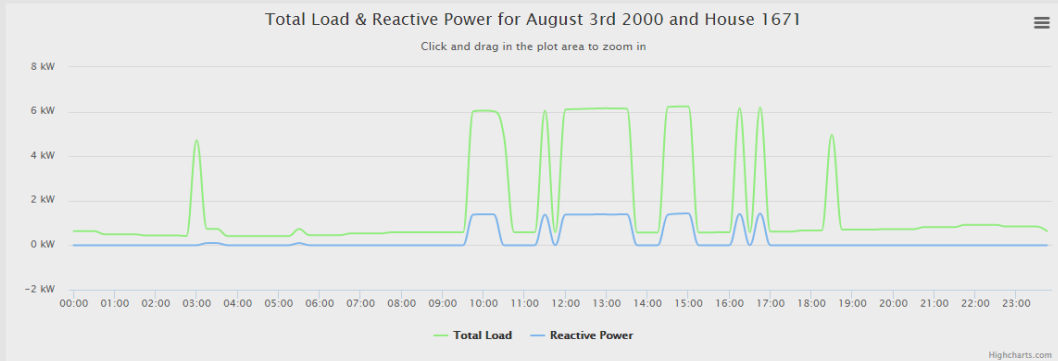


Figure 54 Line Diagram for multiple houses of the Houses table.

Spider Chart for 03-08-2000 for node 671

You can also see the data for House in Line Diagram, pressing the button below.

Line Diagram Spider Diagram

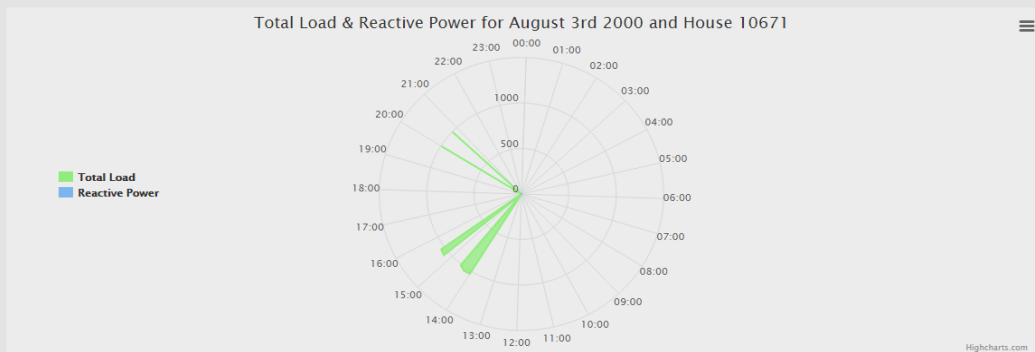
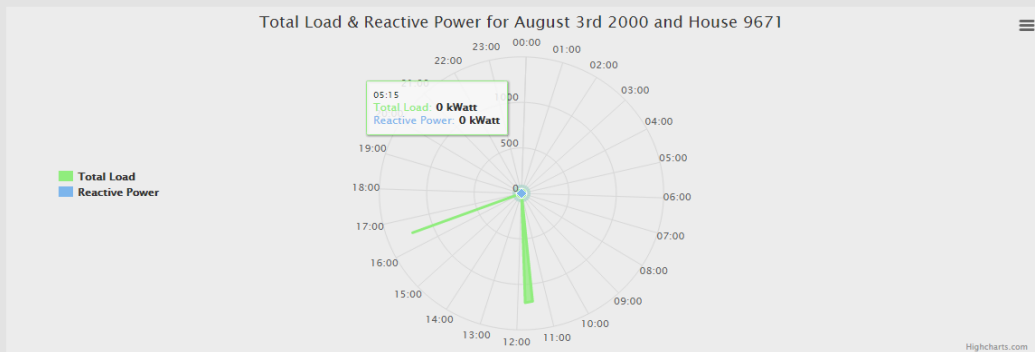
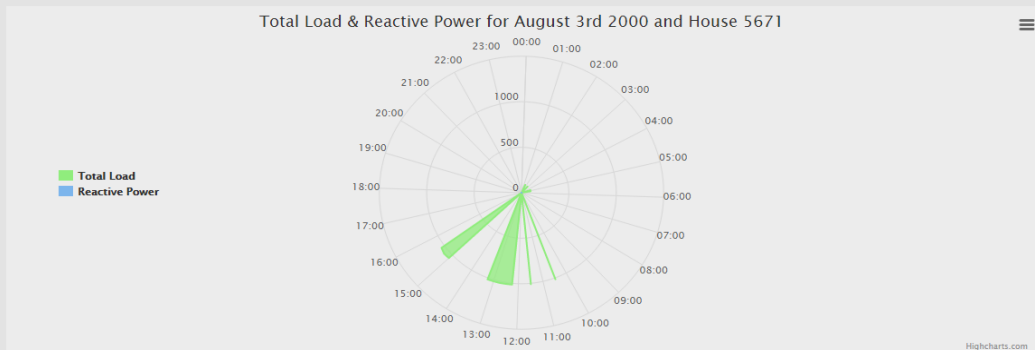
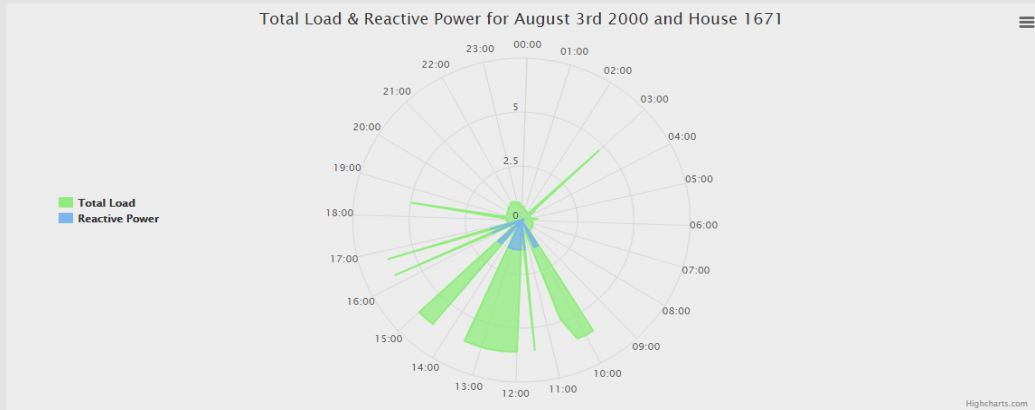


Figure 55 Spider Diagram for multiple houses of the Houses table.

Market Pool

A very important table for our web application is the “Market Pool” table. In this table the user can view and analyze how the values of price and load change over time and per each day. The market pool table depends only on Date, as it includes total information of the system.

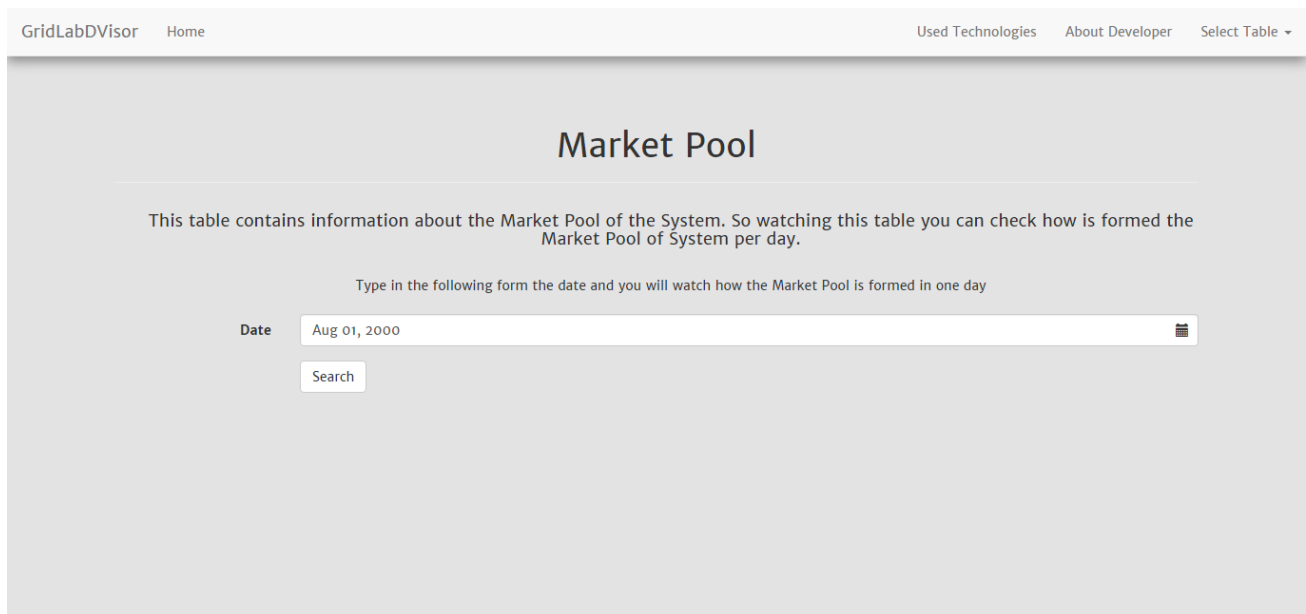


Figure 56 The selection page for Market Pool table.

In this table the user can view the average price of the energy on a given day, as well as the power quantity that the seller sold and the buyer bought. This table also contains the clearing quantity of the power, which is the sum of all of the buyer’s quantities for which the bid was higher than the marginal seller’s bid price. The price of the energy is defined by the demand of energy by consumers at any given time. The more energy is required, the more its price rises and vice versa; when there is not much demand for energy, its price remains low.

For example, as we can see in the following figures, the elations of price happen at noon, between 15:30 and 17:00, and at night between 21:30 and 23:30. The rest of the day the energy price remains low.

For this table, we decide to use Line and Spider charts, because they are suitable for this presentation.

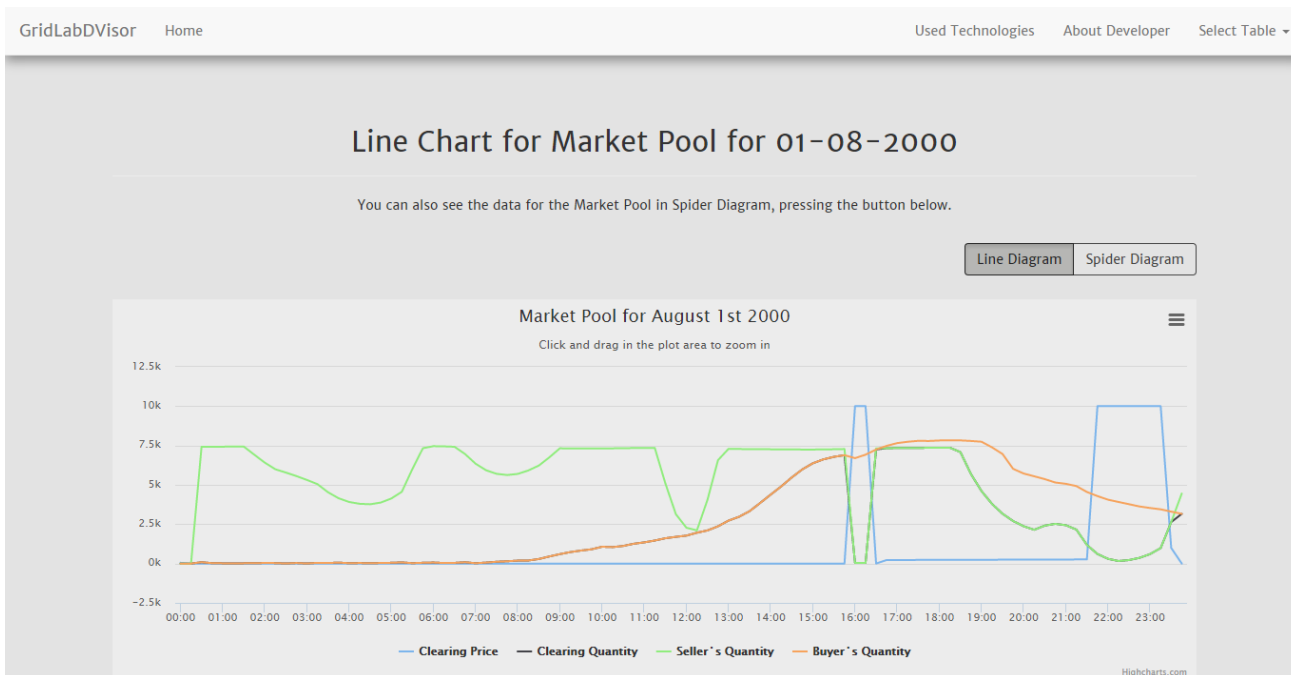


Figure 57 The Line Diagram for Market Pool table.

We chose these diagrams because they aid the user in discovering the best time to ask for energy, depending on the pattern of energy prices.

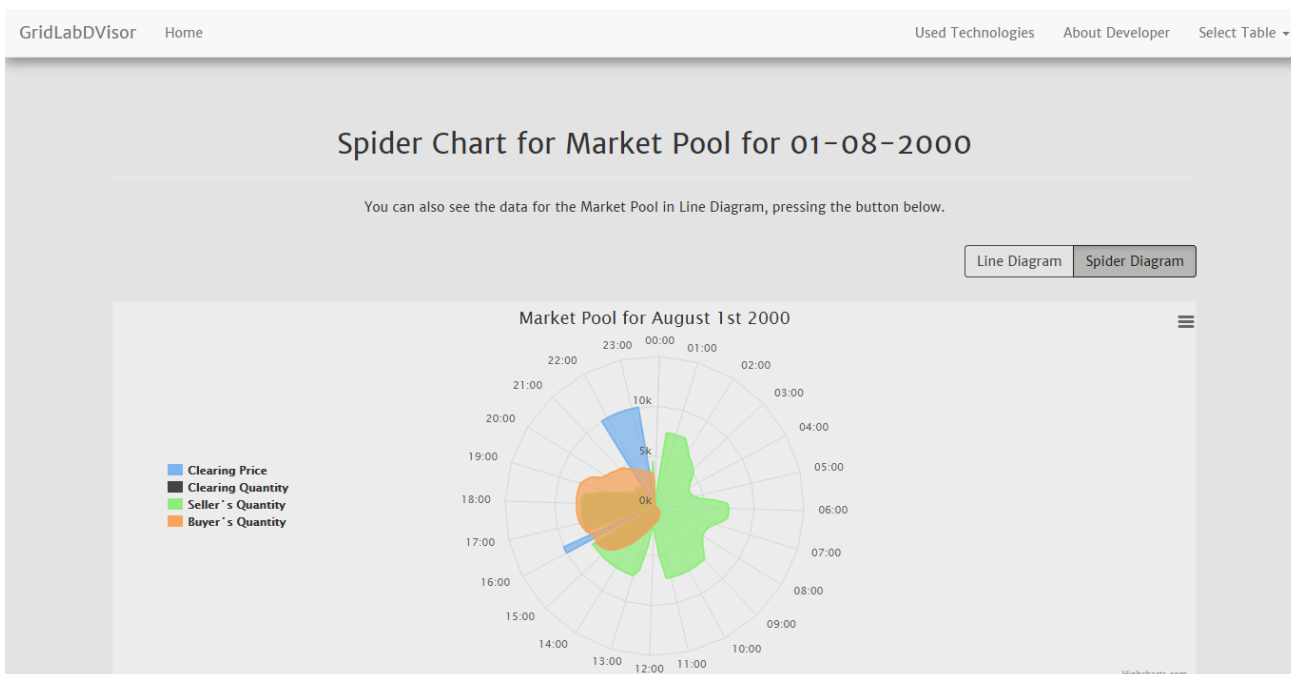
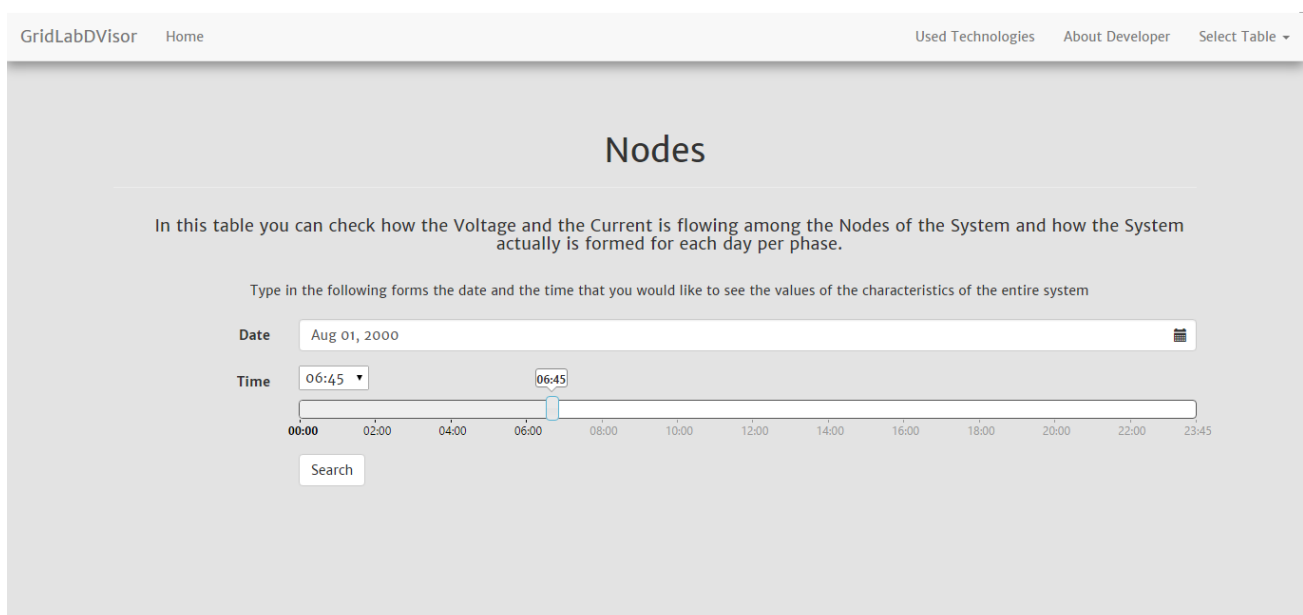


Figure 58 The Spider Diagram for Market Pool table.

Nodes

One of the most important tables of our system is the “Nodes” table. In this table the user can select for which time quarter of a specific day they would like to see the power flow among the nodes of the system. The selection form of this table has only the “Date” input, which we described in a previous paragraph and the “Time” input. “Time” input is actually a slider with which users can select the quarter for which they wish to review the power flow of the system. The user can select a particular time either by clicking on the input field and the dropdown window that appears or simply by sliding the needle over the axis.



GridLabDVisor Home Used Technologies About Developer Select Table ▾

Nodes

In this table you can check how the Voltage and the Current is flowing among the Nodes of the System and how the System actually is formed for each day per phase.

Type in the following forms the date and the time that you would like to see the values of the characteristics of the entire system

Date

Time

Figure 59 The selection page of Nodes table.

By pressing the “Search” button on the select page, the user will automatically view the topology of the nodes of our system. Each node of the graph has a voltage value and each edge has a current value. Both voltage and current include all three phases (A, B and C) and they are in complex form. The voltage is defined per system & graph node, while the current is defined per graph edge, which is the flow between two nodes of the system. The chart we chose for this table, shows the connection between the nodes of the system, as well as the values of power of each node. We decide to present values of this table in complex form, in the total graph and in real form in the “per phase” presentation. We also decided to present this chart both in hierarchical form, like a tree, and in the unsorted form.

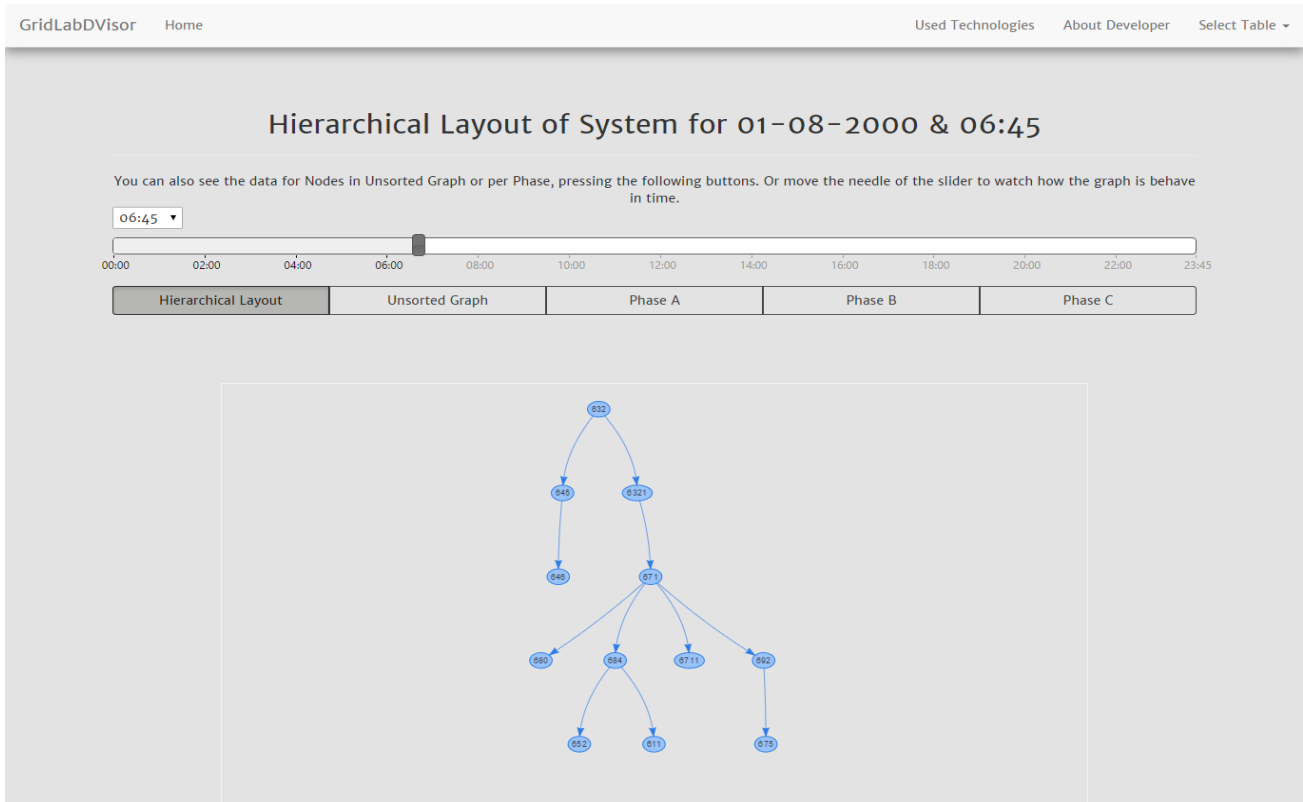


Figure 60 Hierarchical Layout of the Nodes table.

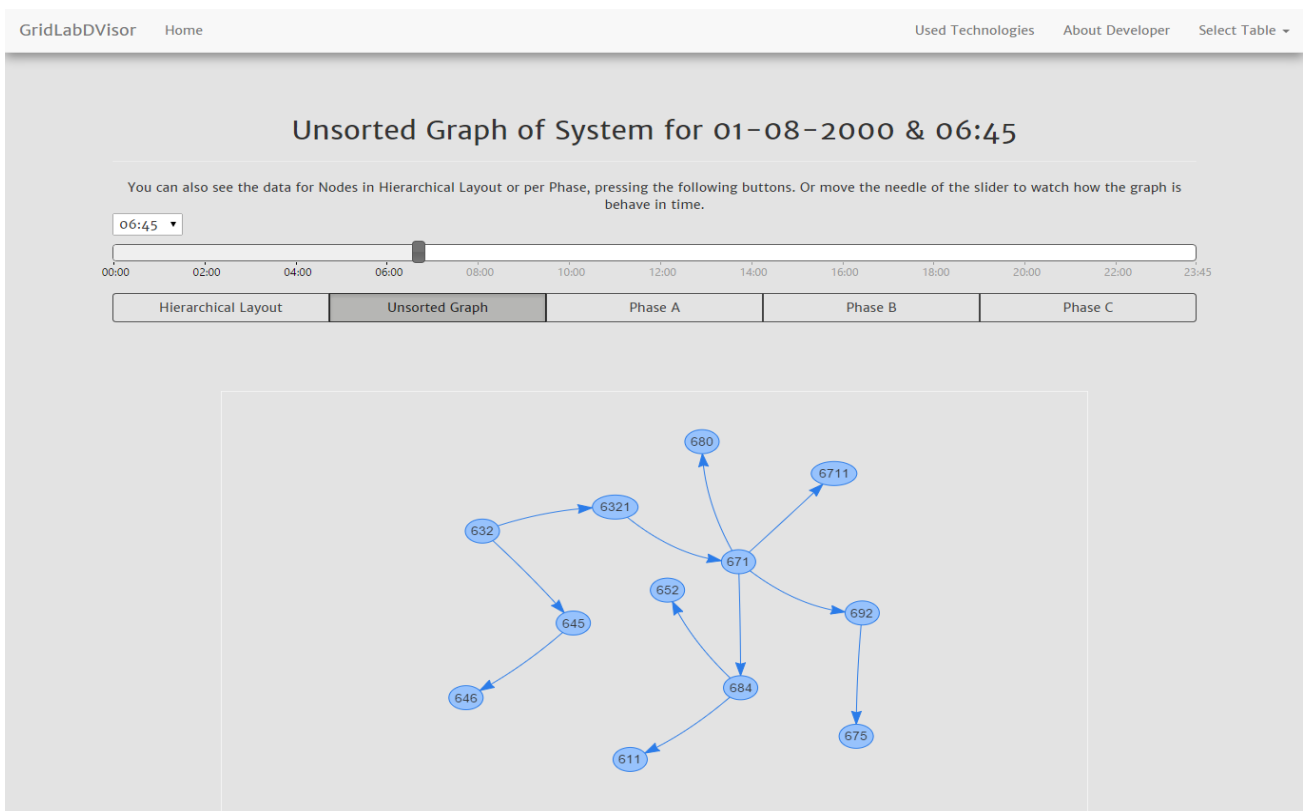


Figure 61 Unsorted Graph of the Nodes table.

On the top part of this page a time slider exists. When the user moves the needle of this time slider, the time is updated instantly and the chart for the new time is loaded and appears below. Sliding the time slider's needle the user is able only to change the time for the presented graph and not for the day.

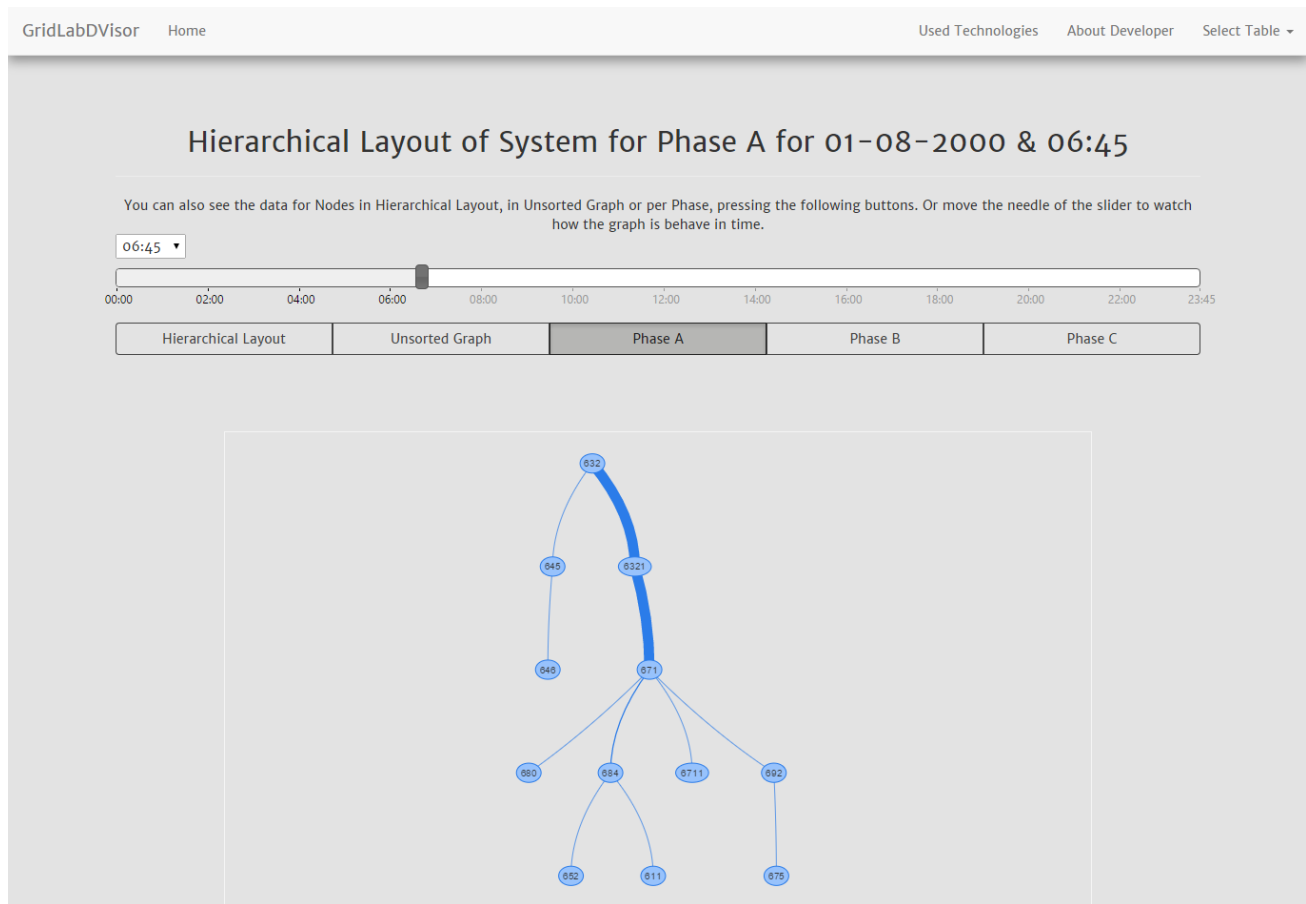


Figure 62 The Hierarchical Layout of phase A of the Nodes table.

Transformer

Another table, which is very important for our system, is the “Transformer” table. In this table, as we can understand by the name of the table, we have information about the unique transformer of the system. The information this table contains is the power that flows into the transformer, the power that flows out of it and the power losses it probably has. This information is different for each phase of the system. That’s why we developed three charts, one for each phase. Each chart has the power that flows into

the transformer's phase, the power flows out of it and the power losses it probably has. These numbers are in complex form. In the selection form of this table the only thing the user has to select is the date and the time for which they would like to see the transformer's behavior. The selection form for the Transformer is just like the corresponding form in the Nodes table, which we have already analyzed.

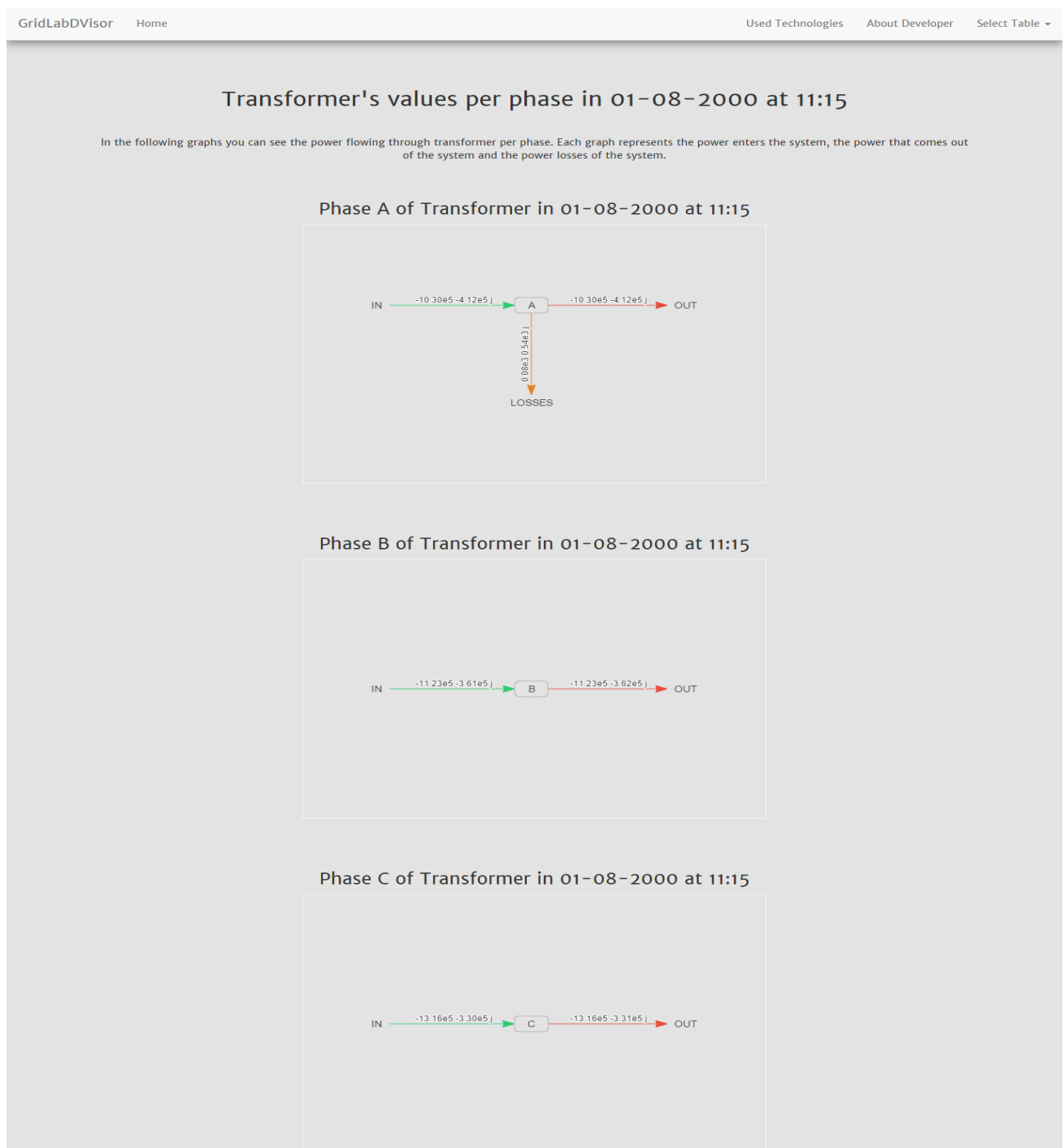


Figure 63 The charts per phase in the Transformer's table.

System Graph

The last, but not least, view of our web based application is the “System Graph”. When a user navigates to it, the graphic topology of the whole system appears. It does not depend on any specific parameter, like the date or the node, because it is a global diagram of the system and it does not change. For this reason there is no need for the existence of a selection page.

The following graph shows the connection among the nodes of the system. Our system has 10 basic nodes, where each of them has a specific number of houses. In our graph we also show all the houses contained by each node. Using the zoom-in function the user can also see the id of the chosen node or house.

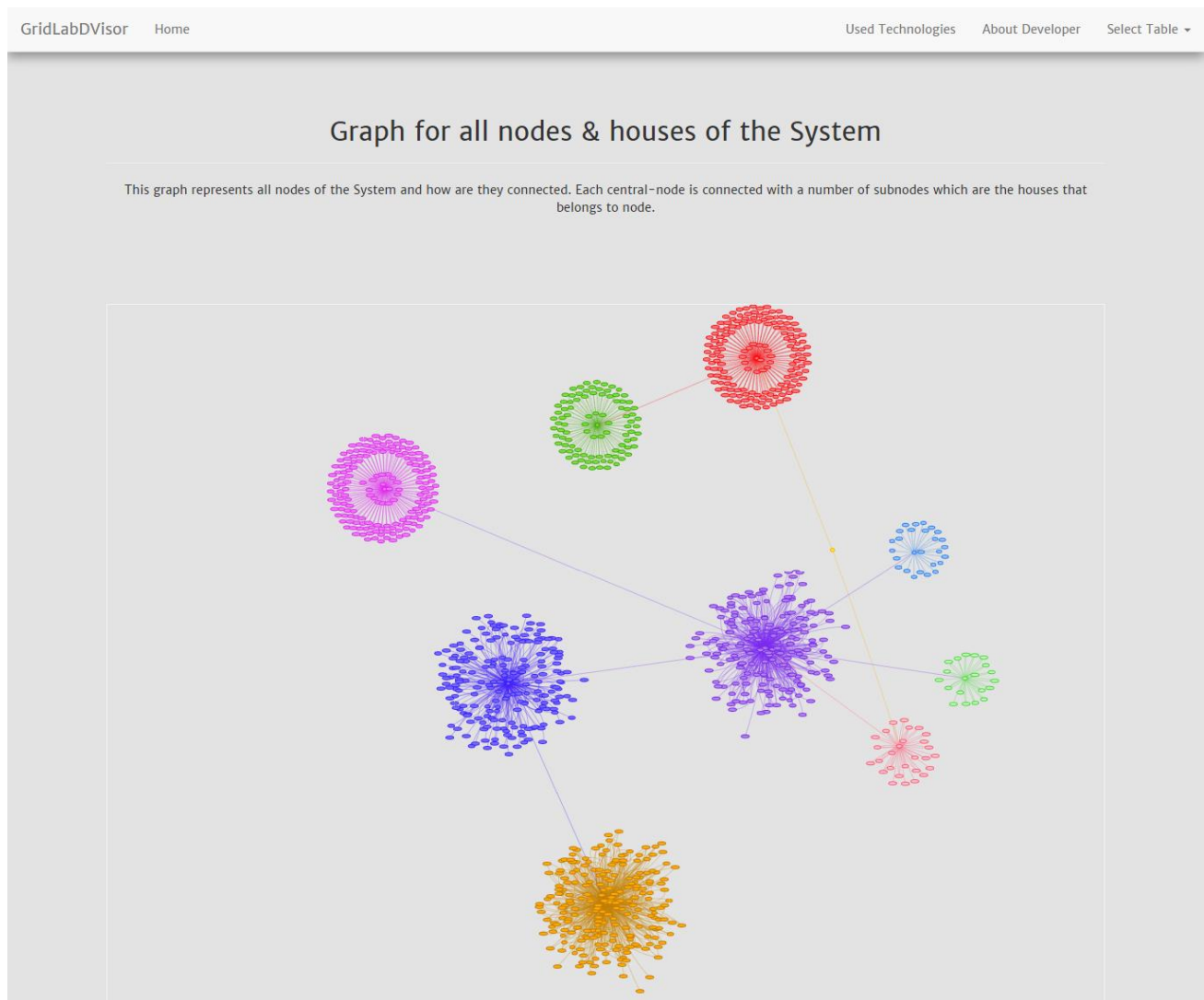


Figure 64 The System Graph of our application.

Chapter 5

5. Differences with existing tools

GridLAB-D is a new power system simulation tool that provides valuable information to users who design and operate electric power transmission and distribution systems, and to utilities that wish to take advantage of the latest smart grid technology. [27]

During a survey we conducted, we did not find any web application that consumes and analyzes the results of the GridLAB-D. So, we decided to develop the **GridLab-D Visor**.

GridLab-D Visor is a web based application, which uses data derived from GridLAB-D's experiments and through suitable visualization techniques, presents them to the user. We do not exclude the possibility of existing similar applications, which we have not found during our research. If there is such an application, then it likely was developed for educational reasons, like ours.

In research we conducted, we could not find web applications for visualization of GridLAB-D's data, but we found some visualization tools which are incorporated by GridLAB-D simulation platform and operating presentations of the data within the platform.

In this chapter of our thesis project we will list some of the most basic differences between existing tools for visualization used by GridLab-D and our project.

The main difference between GridLAB-D's visualization tools and our project is that our project has been developed as a web application accessible to from the Internet, while the others are Desktop applications, accessible only users that have installed them on their machines.

We decided that our application should be easily accessible by users, without them having to download and install files and programs on their computers. A user of our web application can easily and quickly see the graphic presentation of the desired variable, simply by opening a web browser and typing a URL address in the address bar. When users navigate to homepage of our application, they can easily find their way towards our various internal views and tools guided by simple descriptions and instructions. On the other hand, existing visualization tools for GridLAB-D are files, which to be used must be download to a user's computer, while the GridLAB-D installation is a prerequisite. Which means for someone to use these visualization

tools, they must first have installed a list of programs on their computer, like GridLAB-D, Microsoft Visual Studio etc.

Following the previous argument we should keep in mind that the GridLAB-D is a system simulation platform which the average user will not be able to use to get the results they want. And why should they have to?

Users deciding between web based and desktop solutions, should also base their decision on the system requirements imposed by each one. A web based solution, requires only a web browser, that is fairly simple to install and not particularly resource taxing. Tools like the GridLAB-D and the Visual Studio are quite demanding in terms of CPU, memory and operating system, and an average personal computer running on a Windows OS may not be able to satisfy these demands.

All the problems mentioned can be managed if instead of a desktop software we use a web application. With a web application we bypass all the previous problems, such as the lack of memory and incompatibility of operating system as we only need a browser to see the information we like. Moreover, we can assume with certainty that an average user knows how to use a browser.

Chapter 6

6. Synopsis & Future Work

6.1 Synopsis

The objective of this thesis project was the development of a web based application, which receives data from the GridLAB-D - a power system simulation tool - and through appropriate processing and suitable visualization techniques achieves the visualization of given data, in such a way that patterns can be spotted quickly and easily by the user of the application.

Our main goal was to present the raw and valuable data coming from GridLAB-D in an intelligent and efficient manner using data visualization techniques, so that the users are able to review the data they want at any given time, without doubting their correctness or accuracy. The data format is selected each time by the user from a set of supported formats.

Finally, our system is designed in such a way that it can manage Big Data. The amount of data given to us, was sizable, but not huge. We developed our web application in a way that it can handle much larger amounts of data, provided that they are in the same format.

6.2 Future Work

As for future work, a possible extension we are planning, is to add extra functionality to our system. To be more specific, we consider it is very important to add map charts in our implementation. And that's because a map can provide further clarity and prestige to our application. Import maps in our web application will be a great expansion. However, we did not include them in the original version because there was not enough time.

Another possible extension of our system could be to convert it to a real-time system. This means that we could get data directly from the server of the GridLAB-D and not use our intermediate server. But to achieve that, GridLAB-D has to develop an application programming interface (API), which would give us access to its data.

I think that with these extra functionalities, our web application would be more effective and useful.

Chapter 7

7. References

- [1] <https://en.wikipedia.org/>
- [2] http://www.sas.com/en_us/insights/big-data/data-visualization.html
- [3] <http://www.systems-thinking.org/dikw/dikw.htm>
- [4] <http://raphaeljs.com/>
- [5] <http://visjs.org/>
- [6] <http://www.graphdracula.net/>
- [7] <https://jsplumbtoolkit.com/>
- [8] <http://sigmajs.org/>
- [9] <http://js.cytoscape.org/>
- [10] <http://polymaps.org/>
- [11] <http://openlayers.org/>
- [12] <http://kartograph.org/>
- [13] <http://ihere.net/>
- [14] <http://metricsgraphicsjs.org/>
- [15] <https://square.github.io/cubism/>
- [16] <https://github.com/square/crossfilter>
- [17] <http://code.shutterstock.com/rickshaw/>
- [18] <http://www.navicat.com/>
- [19] <https://nodejs.org/en/>
- [20] <http://expressjs.com/>
- [21] <http://sailsjs.org/>

- [22] <https://www.mysql.com/>
- [23] <https://www.npmjs.com/>
- [24] <https://jquery.com/>
- [25] http://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm
- [26] https://developer.chrome.com/apps/app_frameworks