

Czech University of Life Sciences Prague
Faculty of Economics and Management
Department of Information Technologies



Diploma Thesis

Using React framework in dashboards

Author: Fati Sylqa

Supervisor: Ing. Milos Ulman Ph.D.

© 2020 CULS Prague

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Dipl.-Ing. Fati Sylqa, BSc

Systems Engineering and Informatics
Informatics

Thesis title

Using React framework in dashboards

Objectives of thesis

The main objective of the thesis is to evaluate performance of React framework in a web application based on dashboard.

The partial goals of the thesis are:

- To make a literature review of the current state of development frameworks for web applications with a special attention to dashboards.
- To develop a sample dashboard in React framework.
- To test the dashboard performance and compare with other development frameworks.

Methodology

Methodology of this research will consist in reviewing the books, articles and documentation of react framework and other JavaScript frameworks.

Also we will try to look for differences that those JavaScript frameworks has on web applications comparing to React Native framework.

The proposed extent of the thesis

60 – 80 pages

Keywords

React, JavaScript, Web, Dashboards, Framework, Client Side

Recommended information sources

- Angelos, Chalaris. 2018. Javascript Framework Comparison with Examples (React, Vue & Hyperapp). Hackernoon.com. [Online] Jul 11, 2018. <https://hackernoon.com/javascript-framework-comparison-with-examples-react-vue-hyperapp-97f064fb468d>.
- Eeda, Naresh, "Rendering real-time dashboards using a GraphQL-based UI Architecture" (2017). Electronic Thesis and Dissertation Repository. 5136
- Lebensold, Jonathan. 2018. React Native Cookbook: Bringing the Web to Native Platforms. Sebastopol : O'Reilly Media, Inc., 2018. 1491993790, 9781491993798.
- Niclas Hansson, Tomas Vidhall. 2016. Effects on performance and usability for cross-platform application development using React Native. <http://www.diva-portal.org/>. [Online] June 16, 2016. [Cited: Feburary Saturday, 2019.] <http://www.diva-portal.org/smash/get/diva2:946127/FULLTEXT01.pdf>. LIU-IDA/LITH-EX-A-16/043-SE.
- Trends in Web Based Cross. Smeets, Ruben and Aerts, Kris. 2016. Issue. 6, s.l. : IJCSMC, 2016, International Journal of Computer Science and Mobile Computing, Vols. Vol. 5,, pp. 190-199. ISSN 2320-088X.

Expected date of thesis defence

2019/20 SS – FEM

The Diploma Thesis Supervisor

Ing. Miloš Ulman, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 26. 8. 2019

Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 14. 10. 2019

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 27. 03. 2020

Declaration

I declare that I have worked on my diploma thesis titled "Using React framework in dashboards" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on: 06.04.2020

Fati Sylqa

Acknowledgement.

“In the name of God, the Most Gracious, the Most Merciful”.

All the praises and gratitude due to Almighty God who gave me the energy and focus to finish the writing of the diploma thesis. A special honest appreciation goes to my supervisor Ing. Milos Ulman Ph.D. who helped me in every phase of the thesis with such a great commitment and correctness

Nevertheless, many big thanks go to all my family, my girlfriend, and my friends for giving me unconditional support during the whole time of the studies.

Using React framework in dashboards

Abstract

The importance of the JavaScript framework in web development is very crucial and fundamental for building single-page applications. When we want to develop an application that requires a lot of data interaction and data visualization like a web dashboard, the challenge for finding the most suitable, flexible and lightweight JavaScript framework is very demanding.

This paper tries to compare the top JavaScript framework in the aspect of usage and data entry operations through a benchmark script and to choose one of them which resulted in React as the most efficient framework for developing a single page dashboard.

The case study that was presented in this paper shows the enormous effect that react have with the integration of the data visualization libraries for developing a fast data-driven single page dashboard with interactive maps.

Nevertheless, a lot more work let to do in the future for being able to follow the latest trend in the development of web applications, with special focus on the data visualizations that are being applied in every field of technology, medicine, and economical departments.

Keywords:

JavaScript frameworks, web dashboards, react, web technologies, single page application, data visualization, client-side

Table of Contents

1	Introduction	10
2	Objectives and methodology	11
2.1	Objectives	11
2.2	Methodology	12
3	Literature review	13
3.1	What is react framework.....	13
3.2	Features of react	14
3.3	Comparison of JavaScript frameworks	16
3.4	Developing a dashboard application	21
3.4.1	Difficulties and problems from user’s perspective.....	22
3.4.2	Difficulties and problems from developer’s perspective	22
3.4.3	Impact of JS framework on dashboard development	23
3.4.4	Server-side approach	25
3.4.5	D3.js (Data-Driven-Documents)	26
3.4.6	Integration of D3 with React	26
3.5	Principles of web application development	28
3.5.1	Use case	30
3.5.2	Personas.....	31
3.5.3	Wireframes	32
3.5.4	Prototypes.....	33
3.5.5	User experience	34
3.6	Dashboard in desktop applications	35
3.6.1	Tableau.....	35
3.6.2	Power BI	36
3.6.3	Excel	37
4	Practical part.....	38
4.1	RAIN.....	39
4.2	Tools	40
4.2.1	Balsmamiq.....	40
4.2.2	Visual code	40

4.2.3	Storybook	40
4.2.4	Bootstrap.....	40
4.2.5	Jest	41
4.3	Wireframes of RAIN.....	41
4.4	Impelementation.....	43
4.4.1	Storybook components	43
4.4.2	Code implementation of React components	44
4.4.3	Demonstration of RAIN dashboard application	54
4.5	Testing	56
5	Result and Discussion	59
5.1	React and single page dashboard.....	59
5.2	Comparison of React and other JavaScript frameworks	60
6	Conclusion.....	62
7	Bibliography.....	63
8	Appendix.....	66

List of pictures

Figure 1	JSX (Chand, 2019).....	14
Figure 2	Virtual DOM (Chand, 2019).....	15
Figure 3	(History of React, Vue, Angular) (Daityari, 2020).....	17
Figure 4	(Differeces of vue, react ,angular in GitHub repositories (Daityari, 2020).....	17
Figure 5	Table operations benchmark (Chalaris, 2018)	18
Figure 6	Startup metrics benchmark (Chalaris, 2018)	19
Figure 7	Memory allocation benchmark (Chalaris, 2018).....	20
Figure 8	Data visualization of air pollution (Moraga, 2019);	21
Figure 9	Survey from stack overflow (2019)	24
Figure 10	(WDLC Phases)	28
Figure 11	Example of register and login	30
Figure 12	Persona example (Creating and Using Personas in Software Development: Experiences from Practice, 2014)	31
Figure 13	(Prototype example) (Adiseshiah, 2017)	33
Figure 14	Wireframes of login in and sign up	41
Figure 15	RAIN dashboard wireframe.....	42
Figure 16	Storybook component	43
Figure 17	GeoJson Map	48
Figure 18	Cluster Map.....	50
Figure 19	Line Chart	51

Figure 20 Company table with pagination	53
Figure 21 Sign up & Login form.....	54
Figure 22 RAIN Dashboard.....	55
Figure 23 Rankings of front end frameworks (Raphaël, 2019)	60
(Wattenberger, 2019)Figure 24 Changes of JavaScript ecosystem during the years	61

1 Introduction

Today, the influence of the web development interface is very big and it is evolving rapidly. The main factor for this growth is the flexibility and simplicity of JavaScript frameworks, which are giving the UI a new multidimensional concept. One of the most famous JavaScript frameworks for UI is React which is used for developing and operating the dynamic User Interface of the web pages with high-income traffic.

Comparing to the other JavaScript's framework react is faster in updating and rendering web applications through the involvement of Virtual DOM. This makes react very compatible to develop interactive dashboards with high volume data.

The trend of javascript languages is rising year by year, and react is still maintaining the leading position as Js framework in the market. (dhtmlx, 2019)

Since the last decade, dashboards are used everywhere as a standard business tool in representing the data that the user was interested to see in a brief image. As we can see they are a crucial part of every web application that dynamically processes big sets of data. To specify the design of today's dashboard compare with the traditional one whom they were focused only on showing the information altogether, now can see that dashboards are more lightweight, components are mapped beautifully and the flowing of the data is structured and asynchronous.

Starting from the rising trend of JS during the last years in web development technology, we are going to see how the dashboard implementation will be more minimalistic design and functional processing of information

2 Objectives and methodology

2.1 Objectives

The main objective of the thesis is to evaluate performance of React framework in a web application based on dashboard.

The partial goals of the thesis are such as following.

First, a literature review of the current state of development frameworks for web applications with a special attention to dashboards will be done. The literature review will consist in exploring the top JavaScript frameworks and how they operate in developing embedded user interfaces, within the nature of dashboards applications

- To develop a sample dashboard in React framework. Developing the dashboard will include: creating interactive tables and maps that are going to show how company data are distributed and changed during the years in America

To test the dashboard performance and compare with other development frameworks.

Testing and comparing phase will go through a benchmark JavaScript tool, which will use automated benchmark driver to measure different rendering operations that happens in dashboards concretely in tables. This testing benchmark will include different JavaScript frameworks like: React, Vue, Angular and Hyperapp.

2.2 Methodology

Methodology of this research will consist in reviewing the books, articles and documentation of react framework and other JavaScript frameworks. As far as our practical part will be implementation of geodata visualization web dashboard, main key will consist in researching literature for handling geospatial data through JavaScript framework. The other part will include how those data are going to be integrated with React.

Also, we will try to look for differences that those JavaScript frameworks has on web applications, comparing to React. We will go through the advantages and disadvantages of each top JS frameworks, to get a better perspective in choosing the ideal framework for our type of web dashboard application. Special focus will be: using JS benchmark script for testing JavaScript frameworks in different use cases.

3 Literature review

In the following subchapters, we are going to look for main things that makes JavaScript so special in creating a data-driven application like a dashboard. The main focus will be on researching and comparing top JS frameworks in creating of data visualization dashboard.

3.1 What is react framework

React is JavaScript front-end library that is used for building interactive UI (Chand, 2019). It was founded by Facebook and its used by Instagram, WhatsApp and many other big companies.

The idea of developing React was to solve the complexity of user-interfaces with big datasets that changed during the time. So the engineers at Facebook decided to build something that is state manageable and handles data in one-direction. (Gackenheimer, 2015)

The structure of react is based on set of different components like smart and static components. In React every UI Is built from components, so there is no mismatch of multiply types to worry about (Johnny, 2017). The approach of react is asynchronous with the involvement of virtual DOM, which give user-interface faster response to events without needing to reload full page.

React can also be used as an open source library for developing mobile application in : IOS, Android and UPW. This mobile version which uses the same principles of react;s javascript library is is called React Native.

The principle that react wanted to show is: “learn once, write anywhere”.

3.2 Features of react

The main features that made react unique are depicted in Figure 1.

1. **JSX**- is a combination of JavaScript and HTML which gives opportunity to embed DOM into JavaScript code. So it's like a template engine for React Js, for wrapping and compiling the structure of the code. (Gackenheimer, 2015)

One definition for JSX specification is described by facebook in the following sentence:” ... is to define a concise and familiar syntax for defining tree structures with attributes. A generic but well defined syntax enables a community of independent parsers and syntax highlighters to conform to a single specification.” (2014)



Figure 1 JSX (Chand, 2019)

2. **Virtual DOM**¹- has the effect to make minimal changes into html document comparing to real DOM which is more static. Basically, virtual DOM is a representation of user interface which is kept in memory and it is synchronized with “real” DOM.

The virtual DOM makes the changes in HTML structure only in those parts when it is triggered from the react elements. So updates in real DOM structures are sent by batches and not by whole elements, this consist of high-performance rendering and reduce the unnecessary memory usage.

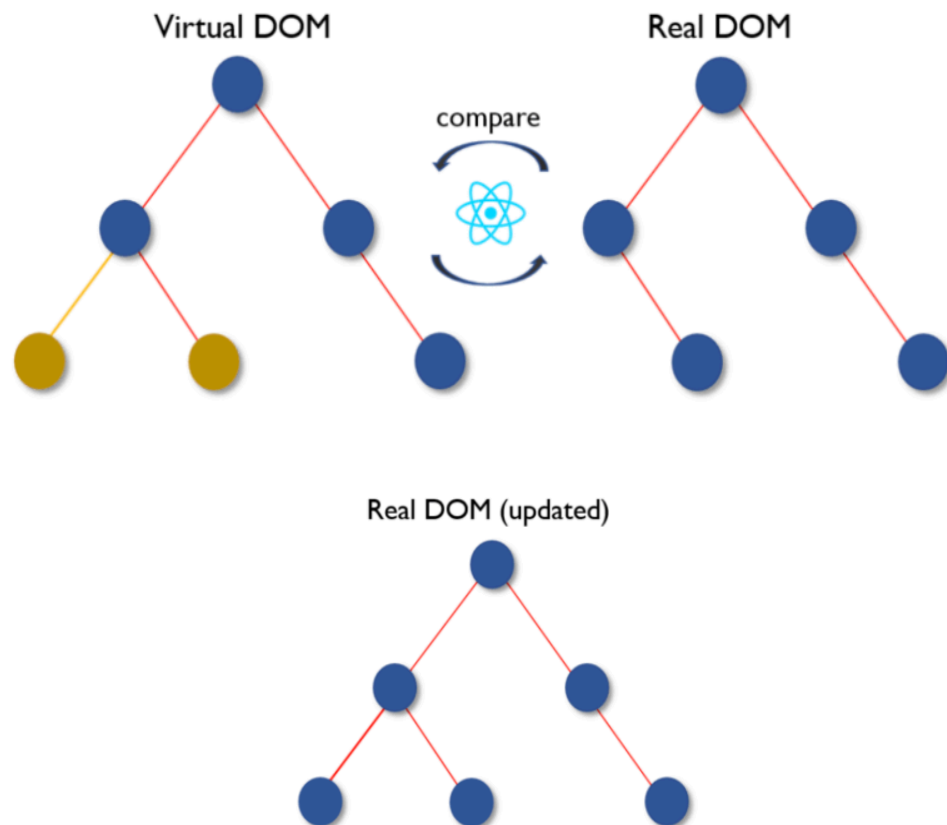


Figure 2 Virtual DOM (Chand, 2019)

¹ Document Object Model is an abstraction for structured text which is used as in-memory representation of HTML text document.

3. **Server-side scripting** – pre render the initial state of the objects and components that are in the page, the main benefit is that browser can run without re-loading the JavaScript files. Users get the page full loaded from one single request to the server, this contribute on transforming web application into the SPA ²
4. **One-way data binding**- means that data flows in one direction from a global or parent state to children components, this contribute on making the handling of the data so much easier to change over the period of time. This approach differentiates React from other JavaScript frameworks who use two-way data binding. What makes those two different is that the one-way data binding changes the UI only if the global state changes, meanwhile the two-way data binding changes both automatically if one of them is triggered.

3.3 Comparison of JavaScript frameworks

The most popular JS framework in today's market are : React, Vue, Angular.

Here we are going to mention some brief history for each of the frameworks (Figure 3):

-**Angular** was developed by google in 2010 is a typescript and MVC framework, is suitable for companies with large teams.

-**Vue** is an open source js framework that is develop from an ex google-employee Evan You. It's has very lightweight library for developing SPA(Single Page Applications).

-**React** was developed by facebook in 2013, is very flexible in terms of integration and migration of different libraries.

² Single-page application stands for web applications that get rewrite the current web pages by getting data in dynamical ways. Usage example of SPA are : google, facebook, twitter etc.

	Angular	React	Vue
Initial release	2010	2013	2014
Official site	angular.io	reactjs.org	vuejs.org
Approx. size (KB)	500	100	80
Current version	9	16.x	2.6.x
Used by	Google, Wix	Facebook, Uber	Alibaba, GitLab

Figure 3(History of React, Vue, Angular) (Daityari, 2020)

In GitHub³ we can see the statistics of three frameworks in aspect of development and support. Both of react and angular has support from their respective company(google and facebook) on the other way vue js depend on open source community. (Daityari, 2020)

The statistics shown on the figure 4 tell us the raising popularity that React have reached in comparison of Angular js and how close is to Vue even though the latest one it's the newest. Both, Angular and React have they community support by they respectable companies(google and facebook), meanwhile Vue is a rising open source community. This tells us that the future of those three js frameworks definitely will not stay the same as it is currently.

	Angular	React	Vue
# Watchers	3.2k	6.6k	6.0k
# Stars	57k	144k	157k
# Forks	15.9k	27.6k	23.7k
# Contributors	1,089	1,361	289

Figure 4(Differences of vue, react, angular in GitHub repositories (Daityari, 2020)

³ GitHub is a working environment for software development which allows to publish and deploy the code. It is open source and everybody can contribute to it.

This article (Chalaris, 2018) describes the comparison of three JavaScript frameworks (react, Vue and Hyperapp). Beside differences in code patterns and in life cycle method, its tries to show the comparison in performance of these frameworks through testing in different subjects like: duration in millisecond of table operations, startup metrics and memory allocation.

Name	react-v16.1 .0-non-keyed	vue-v2.5.1 6-non-keyed	hyperapp-v 1.2.0-non-keyed	vue-v2.5.1 6-keyed	react-v16.1 .0-keyed	hyperapp-v 1.2.0-keyed
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	59.5 ± 1.9 (1.0)	62.4 ± 1.9 (1.0)	66.9 ± 3.5 (1.1)	155.4 ± 3.6 (2.6)	155.6 ± 1.6 (2.6)	152.1 ± 4.7 (2.6)
partial update Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows.	85.6 ± 3.3 (1.0)	163.3 ± 11.5 (1.9)	272.2 ± 23.5 (3.2)	158.6 ± 9.9 (1.9)	85.0 ± 3.3 (1.0)	267.2 ± 22.2 (3.1)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	14.5 ± 4.5 (1.0)	15.7 ± 3.5 (1.0)	19.0 ± 2.6 (1.2)	22.7 ± 2.9 (1.4)	105.1 ± 1.1 (6.6)	119.3 ± 3.0 (7.5)
remove row Duration to remove a row. (with 5 warmup iterations).	36.1 ± 1.1 (1.0)	40.3 ± 1.8 (1.1)	42.6 ± 2.9 (1.2)	56.6 ± 7.2 (1.6)	51.3 ± 3.1 (1.4)	135.4 ± 3.4 (3.8)
create many rows Duration to create 10,000 rows	1,884.7 ± 32.2 (1.3)	1,511.8 ± 49.5 (1.1)	1,415.5 ± 49.3 (1.0)	1,523.3 ± 42.8 (1.1)	1,880.8 ± 34.7 (1.3)	1,426.8 ± 45.4 (1.0)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	257.8 ± 5.1 (1.0)	342.7 ± 17.5 (1.3)	491.1 ± 16.8 (1.9)	340.7 ± 12.8 (1.3)	260.6 ± 5.5 (1.0)	503.2 ± 10.4 (2.0)
clear rows Duration to clear the table filled with 10.000 rows.	179.3 ± 3.2 (1.0)	192.2 ± 7.9 (1.1)	261.7 ± 3.9 (1.5)	193.1 ± 6.8 (1.1)	178.6 ± 1.7 (1.0)	267.4 ± 5.6 (1.5)
slowdown geometric mean	1.04	1.19	1.46	1.49	1.65	2.53

Figure 5 Table operations benchmark (Chalaris, 2018)

Here we can see the results of the benchmark test for each framework in different operational categories. Firstly each framework is divided in two implementation approaches : keyed and non-keyed.

In keyed mode each named framework create 1:1 relationship between data item and DOM node through a key attribute, so any update to the data item automatically will update the node.

In non-keyed mode the change of the data items can modify DOM nodes that were used before from older data. This mode is more faster because it avoids the DOM manipulations that required to remove and add new nodes.

Table operations testing shows us that react was more performant in most of the categories in non-keyed mode especially in partial update which was way better than other frameworks. Whenever in keyed mode the differences were more tighter between the frameworks, even though react was still leading in the categories of clearing and adding rows into the large table.

In the figure(Figure 6)below we can see the startup metrics for each framework, because of the minimalistic size hyperapp is in the advantage of other frameworks, Vue overcomes react by a very small difference. (Chalaris, 2018)

Name	react-v16.1 .0-non-keyed	vue-v2.5.1 6-non-keyed	hyperapp-v 1.2.0-non-keyed	vue-v2.5.1 6-keyed	react-v16.1 .0-keyed	hyperapp-v 1.2.0-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	82.7 ± 1.3 (1.1)	79.6 ± 1.5 (1.1)	74.0 ± 4.1 (1.0)	79.4 ± 1.0 (1.1)	82.4 ± 1.0 (1.1)	76.1 ± 5.8 (1.0)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	20.9 ± 0.5 (1.3)	17.2 ± 0.6 (1.1)	6.6 ± 0.3 (1.0)	17.0 ± 0.5 (1.1)	20.7 ± 0.4 (1.3)	6.5 ± 0.4 (1.0)
main thread work cost total amount of time spent doing work on the main thread. includes style/layout/etc.	183.1 ± 1.1 (1.1)	176.6 ± 6.1 (1.1)	165.5 ± 1.1 (1.0)	177.8 ± 2.0 (1.1)	181.9 ± 2.1 (1.1)	166.1 ± 2.0 (1.0)
total byte weight network transfer cost (post-compression) of all the resources loaded into the page.	266,206.0 ± 0.0 (1.6)	232,024.0 ± 0.0 (1.4)	165,044.0 ± 0.0 (1.0)	232,029.0 ± 0.0 (1.4)	266,195.0 ± 0.0 (1.6)	165,050.0 ± 0.0 (1.0)

Figure 6 Startup metrics benchmark (Chalaris, 2018)

Also at the memory allocation, we can see that hypeapp is at the top because doesn't use so many external sources and its very lightweight compare to the Vue and React which their architecture is more complex and requires much more memory than hyperapp to perform different operations like updating and clearing rows.

In the end, we can conclude that for bigger and complex projects react and vue are more suitable because of their architecture and fast rendering process, compare to hyperapp which is more compatible with a smaller project which requires not so many complex functions.

(Chalaris, 2018)

Name	react-v16.1 .0-non-keyed	vue-v2.5.1 6-non-keyed	hyperapp-v 1.2.0-non-keyed	vue-v2.5.1 6-keyed	react-v16.1 .0-keyed	hyperapp-v 1.2.0-keyed
ready memory Memory usage after page load.	3.3 ± 0.2 (1.3)	3.2 ± 0.2 (1.3)	2.5 ± 0.1 (1.0)	3.2 ± 0.2 (1.3)	3.3 ± 0.1 (1.3)	2.7 ± 0.2 (1.1)
run memory Memory usage after adding 1000 rows.	7.1 ± 0.0 (1.4)	7.5 ± 0.0 (1.4)	5.2 ± 0.1 (1.0)	7.5 ± 0.0 (1.4)	7.1 ± 0.0 (1.4)	5.2 ± 0.1 (1.0)
update each 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	8.1 ± 0.0 (1.4)	7.6 ± 0.0 (1.3)	5.7 ± 0.1 (1.0)	7.6 ± 0.0 (1.3)	8.1 ± 0.0 (1.4)	5.7 ± 0.1 (1.0)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	11.3 ± 0.1 (2.1)	7.6 ± 0.0 (1.4)	5.8 ± 0.1 (1.1)	7.6 ± 0.0 (1.4)	8.4 ± 0.0 (1.6)	5.3 ± 0.0 (1.0)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	4.3 ± 0.0 (1.4)	3.4 ± 0.0 (1.1)	3.0 ± 0.0 (1.0)	3.4 ± 0.0 (1.2)	4.3 ± 0.0 (1.4)	3.0 ± 0.0 (1.0)

Figure 7 Memory allocation benchmark (Chalaris, 2018)

3.4 Developing a dashboard application

Since the beginning of digital era, dashboards have been focal point in summarizing and displaying the track of data in one graphical user interface panel.

One of the best short definition for dashboards would be: “Dashboards are visual display of data used to monitor condition and/or facilitate understanding” (Wexler, et al., 2017).

One of the examples of dynamical data dashboards is this in the following picture, which is showing the air pollution in different parts of the world:

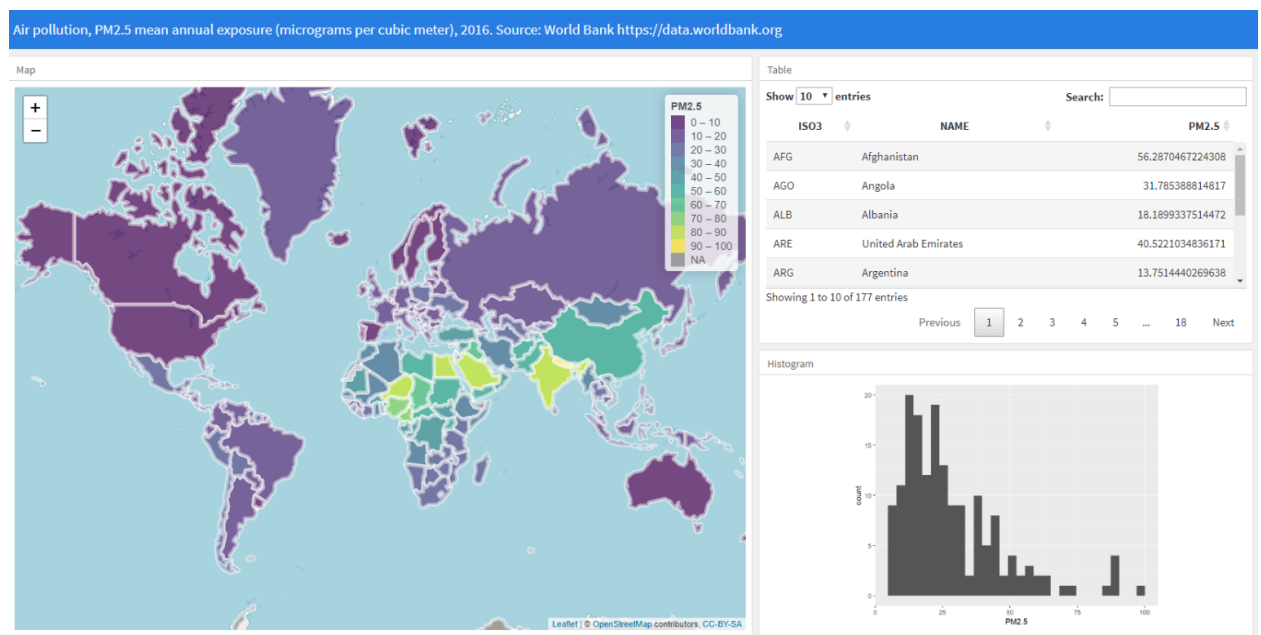


Figure 8 Data visualization of air pollution (Moraga, 2019);

Traditional dashboards used a lot of data metrics, charts and different gadgets in single page without taking into consideration the end user. Evolving of web design has made usability, utility and beauty more unified with each other. This evolution of web affected also the dashboards to be more readable and understandable for end users. (Box, 2015)

The main values that dashboards bring are:

- helping team management to define what things are important
- highlight errors and progress
- common interface for data analysis (Box, 2015)

- who are typical users – what they expect from a dashboard?

The most frequent usage of the dashboards from users is to check the status and prediction of different data that they wish to display. So users mostly want to look for a dashboard that is very structured and informative in summarizing the information that they want.

3.4.1 Difficulties and problems from user's perspective

Users may find complicated if a dashboard is overwhelming and not understandable, also one main point is that dashboards wouldn't be a great source for deep data discovery.

Another problem will be the bad visual representation of the components into the panel of dashboards like small fonts, un-matched contrast between the layers, and chaotic positioning of data elements within the visualization dashboard.

Also, we need to have into consideration the user experience aspect which will lead as to following questions like:

- How much of the dashboard will be user-friendly regarding the intuitive aspect of finding out for the first time how the dashboard works?
- How easy will be for users to navigate through it to find their data? in how many places they have to click? are the tables updatable in real-time?
- Do components of the dashboard are customizable for data searching?
- Is the content of the dashboard structured and understandable for the users, and are the most important information highlighted?

3.4.2 Difficulties and problems from developer's perspective

Building a data analytics web dashboard application can be challenging in finding the right approach of implementing the best patterns that suits the user's need for summarizing the most important pieces of information that they want to see. This leads to finding a way how to refine a big dataset of information into the proper categories.

So the real struggle of developers will be to make a fast data-driven dashboard with a lot of complex information from different sources which requires:

- Handling API requests properly,
- Making the application layout very lightweight
- Designing the structure of the dashboard in a minimalistic approach,
- Finding the right data visualization tools for showing data in disperse way,
- Mapping data with an interactive user interface
- Generating a fully flexible single page application

3.4.3 Impact of JS framework on dashboard development

Before the invention of javascript, traditional web application worked as a client-server architecture when each time the users sent a request to a server, the server responded with a new full requested served web page per each new request that users have made.

This approach changed when was introduced the new version of the client-server architecture was called AJAX. AJAX stands for Asynchronous and JavaScript XML and it is used for loading parts of web pages asynchronously without affecting the rendering of the whole web page. So when a user submits a request thought ajax, the web browser takes this request and processes it as XML-HTTP to server which rendered it to the particular web content that got from the user request. (Paul Colton, 2011)

This makes the web more interactive because it doesn't need to render the whole content of the web instead just the part that is triggered by user requests.

In such matters, the response time of the server is greatly reduced and the flow of data transmission is minimize because all the processing information is done mostly on the client-side.

Meanwhile fot developing dashboards application that is very demanding in the aspect of handling interactive data, ajax is very flexible because it uses HTML, Javascript, and CSS to create dynamical web content in asynchronous approach.

JavaScript as a modern client side programming language is a top technology in developing web applications with big datasets. The big variety of libraries that js contain, makes the development of any data driven application so much fulfilled and efficient for users to develop. (Paul Colton, 2011)

A yearly survey done from stack over flow shows that for seven years in a row JavaScript has been voted as the most popular technology(Figure 9)



Figure 9 Survey from stack overflow (2019)

This shows that for developing web base application like dashboards, JS frameworks are crucial for creating data visualization applications that requires fast data processing with asynchronous approach.

In the following sections, we are going to see more in-depth and elaborated what makes JavaScript compatible, for developing advanced data visualized web applications like dashboard, that represent a different source of information in one user interface.

3.4.4 Server-side approach

The other main thing in development of dashboard is to handle data from database. Javascript has a very nice cross-platform for that called Node js.

Node js- is an asynchronous event-driven JavaScript runtime environment which runs through v8 google chrome engine based. Most languages are based in multi-tasking threads which are waiting one for another process to finish in synchronous way, otherwise node has no threads and relies in callbacks to let you know when the given process is finished (Subramanian, 2017) .

So the main idea of node in web development is to unify client and server side in one JavaScript language.

Express js. – is lightweight web framework for Node js ., so it is based on Node modules⁴ and connect⁵ components – which they are called middleware⁶. For the web applications, express provide MVC(Model-View-Controller) structures, when M(Model) firstly need to be connected with a database like : Mongoose, MySql or Redis. (Mardan, 2014)

Starting from the MVC architecture that's we mention earlier, In express js you can define the API calls that you want to implement in your web application, and render through the node js runtime environment. Also you can define routes and views per each endpoint that you want to show in client-side.

⁴ <https://nodejs.org/api/http.html>

⁵ <https://github.com/senchalabs/connect#readme>

⁶ Middleware is a software that acts like a gateway or bridge between system operation and application in this example: operation system with network configuration.

3.4.5 D3.js (Data-Driven-Documents)

One of the main and the most important JavaScript library that we are going to use to develop the dashboard will be D3.js.

D3.js is a JavaScript open source library that stands for interactive data-visualization that apply data-driven transformation into the documents. It displays data through using web standards like : HTML5 .SVG(Scalable Vector Graphics) and CSS.

It provides developers the ability to create very rich and animated content, that is based on manipulative data using dashboards that contains different maps, tables and charts. (Meeks, 2018)

D3 js is extremely fast and its able to support very big data sets for interaction and data animations (Bostock., 2019), it have all the tools to give you a high-performance data dashboard and sophisticated data visualization. (Meeks, 2018). It handles different data format like: CSV ,JSON and GeoJSON.

3.4.6 Integration of D3 with React

To visualize the data in disperse ways we need to integrate with a front-end library that deals with data driven user interfaces. In this case, React is the most suitable JavaScript library to manage and render d3 operations into the web browser. This, for the reason that react have a very structured hierarchy of data handling and rendering methods like : render, properties(props), state and lifecycle methods. (Meeks, 2018)

- ❖ **Render**- this is one of most used method in react application which returns the elements and methods that are created by React. The rendering process works as a copy of the real DOM called virtual DOM which is mention earlier.
- ❖ **Props**- carries the data from one parent component throught the different children components. So, these props make the changing of the data more easily

- ❖ **State-** initialize the state of the smart components and update the component using method 'this.setState()' which automatically trigger the re-rendering process.
- ❖ **Lifecycle methods-** contains three type of component status: mounting, updating, unmounting . Mounting status initialize the state or the props, updating tell if component is ready to update with certain implemented logic and unmounting detach the component from the process of rendering.

One of the challenges of d3 integration with React is that both of them want to control the DOM, so d3 patterns are in conflict with the virtual DOM of the react.

So for this case most people use react only for the structure of the application, and for data displaying they create containers (usually svg elements) whom they pass to d3, after that d3 create and update the required elements for visualization. (Meeks, 2018)

In the practical part we are going to see how the mapbox platform uses d3 operations for using the the json data and rendering those into interactive maps with different layers and circles .

3.5 Principles of web application development

To structure the development of a web application we need to go through sequential steps that allow us to build a fully implemented project, from gathering the required information to maintaining and testing the application.

For this structured method, we are going through a model called **WDLC(Web Development Life Cycle)** which is based on Software Development Life Cycle Model. (Overview of Web Development Life cycle in Software Engineering, 2018)

WLDC - is a systematic web development approach which splits the implementation part into different consecutive phases like feasibility, analysis, design, coding, testing, implementation and maintenance. Figure 8(WDLC Phases)

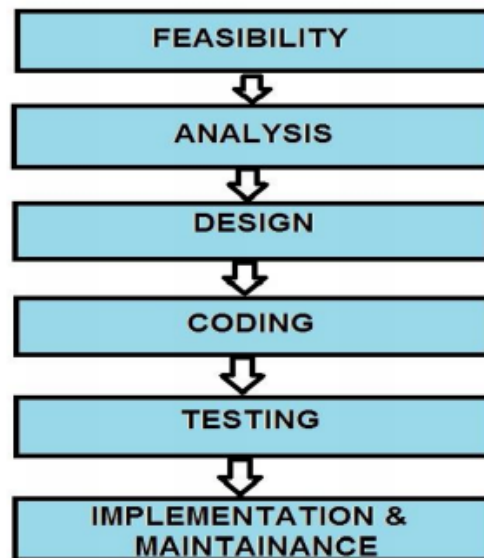


Figure 10(WDLC Phases)

Web feasibility – in this first phase happens the process of gathering information for structuring the plan of the web application. Concretely, targeting the right audience and using the suitable web technologies that the user expects the application to perform.

This phase is very important because it decides the path of other step phases, if something goes wrong in this phase it affects the whole model.

Web analysis- after the data are gathered from different sources, here start the analyzing process of input and output data performance. When the data analysis is finished, the focus falls in that process that meets the required performance from the feasibility phase.

Design and coding – it includes the translation of the conceptual part of the model into the real development of web applications. Developers try to build the application from the results of the feasibility and analysis process. Besides that, they contribute to adapt web design from the user's feedback.

Testing - is the most crucial part in seeing the status of the application from every perspective. Starting from the simplicity and flexibility of web design, to functionality of user interface components like structure of website layout, validity of JavaScript, and functionality of button groups.

Implementation & maintenance – deals with the deployment of the application in hosting server and database. It's the final step of the model which connects to the end-users in real-time. Preserving of application consists of daily maintenance for the latest updates and monitoring each log that happens from users' interactivity..

The main benefits of WDLC model are: being able to adapt complex web application structures and dynamically web content material. Also, this model is very suitable for building different prototype versions and integrate those into any web application.

3.5.1 Use case

Use case modeling try to describe how a functional software process works by using actors and systems to simulate a functionality scenario of the whole process. An actor is a representation of the human interaction role and can exist two kinds of actors: a primary one which initializes a use case and the second one which can potentially participate in the use case in the opposite part of the use case . A use case starts with an input that gets from an actor and communicate sequentially with the system, each iteration starts with a new input from the actor to the system. Usually, the standard use cases start with one actor but can start also with two actors if the complexity of the software application is more bigger. (Gomaa, 2011)

An example of the use case we can see here, the scenario when a user wants to register and to log in into the dashboard web application:

Users can register and login and search and filter data from the database, whenever the administrator can approve the new registration from the user and has the right to edit and delete the user.

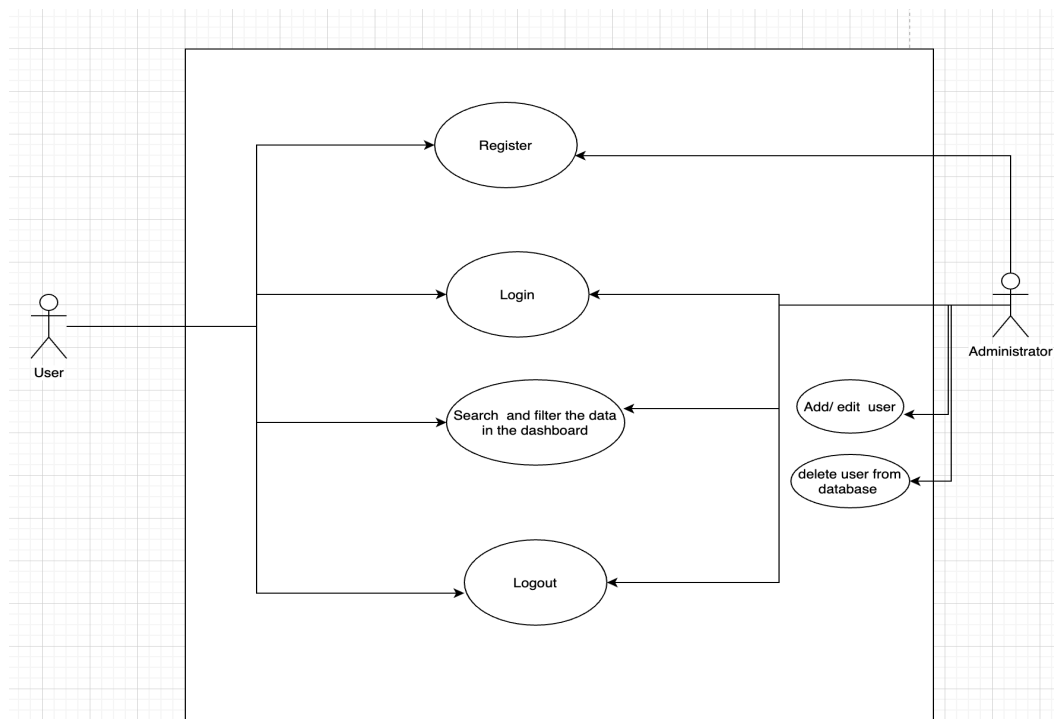


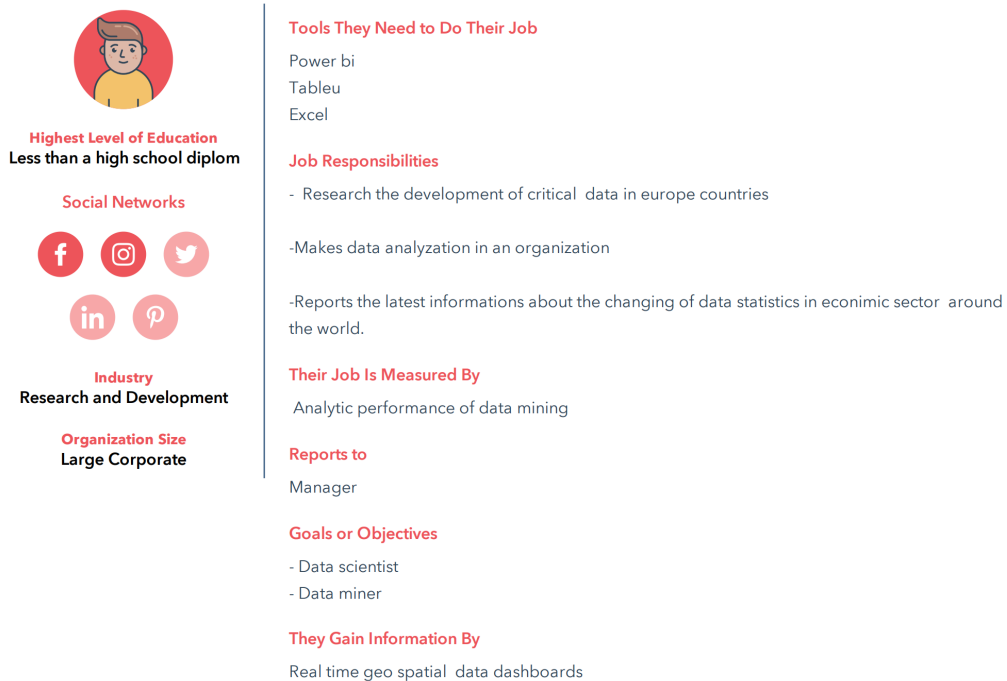
Figure 11 Example of register and login

3.5.2 Personas

Personas is a method that describes a fictional scenario of end-users expertise in a target group. It is used from software developers to have an overlook of user's perspective on their software application. The most common way it's displayed through a photo with a text description of user's personal information and skills.

The benefit of creating personas is that they try to give the developers an awareness that the product is not to be judged by themselves but from the users. This comes from results that sometimes the users do not know what they want, so when something is showing at them it gives more clarity in their demand. Although, this has also some disadvantages in terms of targeting the wrong users when trying to create the persona. (Creating and Using Personas in Software Development: Experiences from Practice, 2014) In the following template we are going to see an example of the persona that is going to be used in our case:

Adem



The persona card for Adem is divided into two main sections. On the left, there is a circular profile picture of a man with brown hair and a yellow shirt. Below the picture, the text reads: "Highest Level of Education: Less than a high school diploma". Underneath that, it says "Social Networks" followed by icons for Facebook, Instagram, Twitter, LinkedIn, and Pinterest. Further down, it lists "Industry: Research and Development" and "Organization Size: Large Corporate". On the right side, there are several categories of information: "Tools They Need to Do Their Job" (Power bi, Tableau, Excel), "Job Responsibilities" (Research the development of critical data in Europe countries, Makes data analysis in an organization, Reports the latest information about the changing of data statistics in economic sector around the world), "Their Job Is Measured By" (Analytic performance of data mining), "Reports to" (Manager), "Goals or Objectives" (Data scientist, Data miner), and "They Gain Information By" (Real time geo spatial data dashboards).

Highest Level of Education
Less than a high school diploma

Social Networks

Industry
Research and Development

Organization Size
Large Corporate

Tools They Need to Do Their Job
Power bi
Tableau
Excel

Job Responsibilities

- Research the development of critical data in Europe countries
- Makes data analysis in an organization
- Reports the latest information about the changing of data statistics in economic sector around the world.

Their Job Is Measured By
Analytic performance of data mining

Reports to
Manager

Goals or Objectives

- Data scientist
- Data miner

They Gain Information By
Real time geo spatial data dashboards

Figure 12 Persona example (Creating and Using Personas in Software Development: Experiences from Practice, 2014)

3.5.3 Wireframes

Wireframes are a simple representation of the initial design that is going to be drawn in the wireframe tool with a white and black paper style. Each page of the web site is sketched by the way of how will look and what will contain. So, it's the initial idea of how the project will look and what feedback clients will give.

Wireframes are usually measured in levels of fidelity- which tell us how close the scope of the user interface design is to be finished (Vegh, 2010)

There are two types of wireframe fidelities:

- **Low fidelity** wireframes are limitless sketches that encompass the general idea of how the application will look, they are done rapidly and try to inform briefly the customers in the layout of the application. The components and window pages that are placed in the wireframe are static without any user interaction. In spite of having limitless functionality, low fidelity wireframes give a clear navigation structure of how the user-interface components will be placed.
- **High-fidelity**- are interactive wireframes that display the blueprint of real application with entry data fields and different button events. They are made to show the real feel of user-interface into the application and are very suitable for user testing. Also, they tend to help very much the developers to be focused only on coding the application and not to make a sporadic correction into the design.

The disadvantages of low-fidelity wireframes are that they are very limited in the detailed specification and in user testing. Whenever high fidelity wireframes are very expensive to develop and takes too much time to be built. (Low vs. High-Fidelity Prototyping Debate, 1996)

3.5.4 Prototypes

The first conceptual part of the development of the application that is base for building and structuring the application are prototypes. Prototypes give a logical plan of how the application is going to be built.

Before drawing in the prototypes is important to bear in mind the requirements that we get from the specifications of the project. Furthermore, it's important to draw a different version of prototypes to get a better scope of the application's functionality.

One example of mock prototype is illustrated in the picture below(Prototype example) when it is shown an advanced visualization dashboard, that contains economic and geographical data.

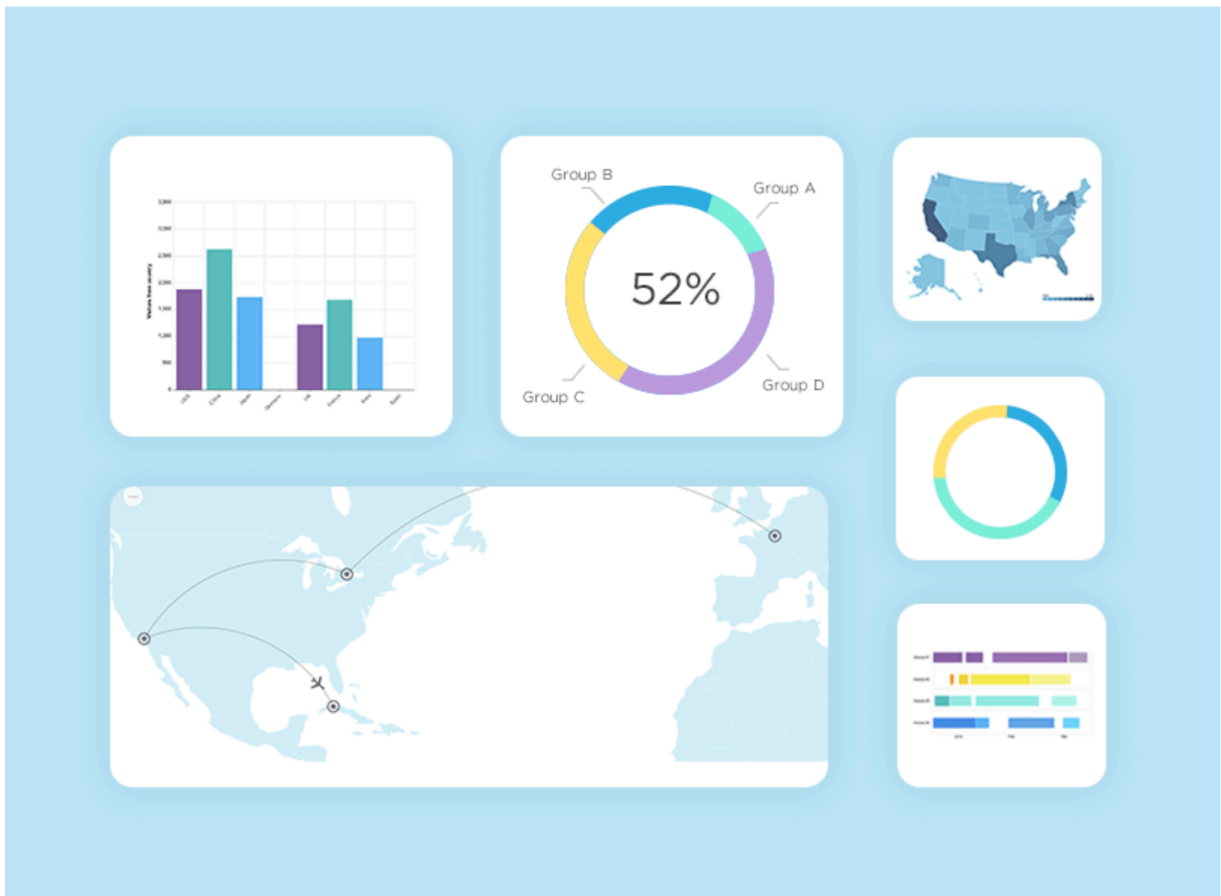


Figure 13(Prototype example) (Adiseshiah, 2017)

3.5.5 User experience

When we try to develop an application like dashboard, it's very important to have in mind the aspect user's perspective in terms of testing and using the application that is target to them. As we know in today's world UX is a buzz word that can be heard in the software design area. User experience or UX is an encapsulation of all the interactions that end-users make with the product or company. It is a very important feedback that the product gets for further improvements in aspect of usability, design, and accessibility.

So UX can be defined as a combination of usability and human interaction. To understand properly this relationship we need to have into consideration the following points (Friedman, 2012) :

- Users don't like to read, they always scan the product
- Users always appreciate the quality content that a product provides before the good design
- If the application doesn't satisfy the expectations that users have, they immediately switch for better product
- Users want to have control over the application without being bothered with unexpected behaviors that the web application may have.
- The web application should be self-explanatory, that allows the navigation of UI to be very intuitive
- The fewer operations that users try in testing an application the better feedback in the first impression.

In general, UX is a good review for testing the application from a different point of view of real human interaction. This helps the development of web application to be more generic and flexible to potential change

3.6 Dashboard in desktop applications

Manifestation of data is very big in every sphere of our digital daily life, for wrapping and analyzing large scale of data with different formats we need monitoring tools like desktop dashboards.

In the following sub chapters, we are going to talk about the most popular data analysis dashboards tools in the market.

3.6.1 Tableau

Tableau is a powerful desktop and cloud application for visualizing data with great analysis and data discovery. It's a business intelligence tool that can be integrated with different big databases like Microsoft SQL, Oracle, Hadoop, etc. So, it doesn't contain a built-in database but is very flexible to connect with third party databases.

What makes this tool very unique among others is that it uses the VizQL (visual query language), which means that we mostly need to drag and drop the data visualization items without being bothered to connect them to data through SQL queries. All this transformation of data query operations goes via this special language VizQL that tableau automates it into a visual representation.

Like we mentioned earlier, the tableau is very flexible in visualizing data analysis, where you can go through interactive maps like filled maps and symbol maps. Having all this into consideration, you can customize very easily interactive dashboards with multiple data schemes and advanced filtering options.

Dashboards from Tableau's perspective are easy-to-use because for combining the visualization tools into one panel, it offers a dashboard framework that creates a new working environment for arranging visualization components by mouse. And after that, you can add particular tableau actions for filtering and highlighting the content of the dashboard. (Milligan, 2015)

3.6.2 Power BI

Power BI is a business analytic tool that stand for data insights and scalable dashboards within the business intelligence scope. It is developed by Microsoft and operates as desktop and cloud application.

Power BI Desktop works as self-service tool that gives users the ability to great powerful data visualization dashboards without depending too much in IT involvement. (Aspin, 2016)

One of the most important features of Power BI is the Power Query which takes the data from different sources(SQL, CSV, web, etc) applies transformation and load them into the dataset of Power BI. This works through an editor that is called Query Editor, which is responsible for loading and selecting different data sources to prepare the data for transformation.

Also, another operation of the Power Query is to shape the data in each column for the user's need, and format the data to make it readable and very accessible to use. Besides this feature Power bi uses other component for modeling and creating the data model - *Power Pivot*, for visualization interactive graphs and maps – *Power View*, and for more immerse 3d maps – *Power Map*. (Aspin, 2016)

The best way that Power BI distribute the personal user content for testing purpose, is via dashboard sharing. This makes possible to share the personal dashboard that the user has with whom he wishes, through providing the email of the recipient and applying two levels of permission: write and re-share.

The usage of Power BI is more in business intelligence experts who deal with analyzing the insight of data flows and visualizing in embedded dashboards. Nevertheless the friendly user-interface with the flexibility of selecting all kinds of data into one combine data set makes the Power BI the adequate choice for data scientists.

3.6.3 Excel

As we have gone through the most usable dashboards application, in the end, we going to talk about the oldest data analytic tool called Microsoft Excel.

Excel was founded by Microsoft in the '80s and it is still used for different data operations, firstly was developed as a desktop application, now it's available also as cloud service.

Excel is a software tool that uses spreadsheets for calculating, filtering and analyzing numeric data. It's a very multidimensional tool that can easily be integrated with other data sources.

In aspects of creating the dashboards, excel has a variety of different combinations tools like pivot tables, pivot charts, pie charts, histograms, etc. All of those can be combined throughout a template of dashboard that excel provides it. One extra thing that can be included in excel dashboard is a new feature called sparkline.

Sparkline is a visualization line tool that was developed to show the trend of the data in minimalistic space. So, for each selection of column data in the table, a sparkline could be customized to visualize the trend or summary of the data in different line graphics.

Like we mention earlier Pivot tables are very crucial for building an effective and data-driven excel dashboard. A pivot table can arrange a lot of different types of categorical data into one summary group of data by processing in different calculated operations. This table can be integrated very well with dashboards because from the interactivity that the pivot table has in updating the view by changing only the data source that directs to it.

It's important to mention after those pivot tables are created excel automatically duplicate the data source and put it in pivot cache memory.

This can give the tables more flexibility to eventual changes but can also double the size of the data. (Alexander, et al., 2013)

4 Practical part

Based on the study of the literature and first objective of the thesis, we formulate first research question:

- 1. *How efficient is React JavaScript framework to build a single page dashboard?***
- 2. *How React JS framework stands in comparison to other JS frameworks?***

After reviewing the comparison of JavaScript frameworks, React was more compatible and up to date with the requirements that the dashboard required. This because of fast data rendering that react is capable to handle through the updating the HTML page efficiently via virtual DOM and fast data connection through the server-side scripting.

As we have discussed in the literature review the importance and differences of JavaScript in dashboard development, here we are going to show the implementation of an advanced data visualization dashboard called RAIN.

In the following sections, we will break down in detail the structure of the rain dashboard with the description of the implementation tools, the displaying of the wireframes sketches and the detailed explanation of the developed code.

4.1 RAIN

The practical project entitled RAIN(Relational Analysis of International Nexus) aims to use advanced data visualization to examine global internalization of firms across the sector in the form of investment projects(ranging from the establishment of international sales office, R&D centers, regional headquarters to mergers and acquisitions) in the context of various macroeconomic, political and other environmental factors.

This dashboard tool will contribute to decision making for different firms around the globe and for entrepreneurs who are interested to invest in different sectors of economic development. Also, it will be used for pedagogical and corporate reasons in the aspect of training purpose to contextualize the international business research and other filed that are related to cross-disciplinary research.

For the moment RAIN is focused only in the United State of America and contains a lot of various economic data that change during the years in different states of the US.

The most typical types of data are GDP per capita, median age, active population, total population, male/female aspect ratio, etc. Also, the dashboard contains data in services, retailing, real estate investments, etc.

All of those data are shown interactively in maps with highlighted polygons per each state, charts will visualize the differences of data between the years, and tables will show the detailed information of each data in a particular column with filtering and searching options.

The RAIN project is being coordinated by the SDSU Center for Advancing Global Bussiness and the Center for Data Analytics and intelligence at CZU Prague. (Ulman & Musteen, 2020).

4.2 Tools

For developing the web application and maintaining the components of the dashboard we used the following tools.

4.2.1 Balsamiq

Balsamiq is wireframe mockup tool that is based on cloud and enable to make minimalistic sketches for designing low-fidelity prototypes. It's very easy to use and has very simple tools to wireframe the ideas from the scratch. Besides operating like a desktop application, it supports also web application with balsamiq cloud which makes very easy to share and review the work between the peers. (Faranello, 2012)

4.2.2 Visual code

Visual studio is a source code editor that is developed by Microsoft and serves as a cross-platform for all operating systems. It's a very lightweight text editor with a powerful code environment that makes it easy for developers to write web, mobile and cloud applications in different languages. Also, it supports very good the lifecycle of the development phase with an integrated git version control engine and builds in debugger. (Sole, 2018)

4.2.3 Storybook

Storybook is an open-source tool for developing and maintaining different user interface components. It allows seeing the design and functionality of every component that is going to be implemented in the application. It can be used by every javascript framework, especially from react that is very handy in developing user interface components.

4.2.4 Bootstrap

Bootstrap is an open-source CSS framework which is used together with HTML and JavaScript plugins. It was developed by Twitter and stands for styling website components like buttons, tables, panels, etc. It is used as a mobile responsive framework and contains a

positioning method called grid system which gives the flexibility to position the elements automatically in fixed places without being bothered about the calculation of the pixels.

4.2.5 Jest

The testing framework called Jest is an open-source JavaScript library that can be integrated into each JavaScript framework. The main functionality of Jest is that is very fast and can cover and handle a lot of code that can be tested with sample testing patterns. The other main benefit is that its free of internal configurations and works independently for every JavaScript project.

4.3 Wireframes of RAIN

The first conceptual plan of RAIN implementation is sketched through a wireframe, which displays the main components of the dashboard operation.

Firstly the application is going to include: **login, sign** and **landing page(dashboard)**.

Login is for existing users and it's going to contain two kinds of account:

- User – has access only in showing the data through different input box filters
- Admin – has the right to import the data through CSV file to the database, also can add and remove users.

Here we can see the wireframes of login and sign in page

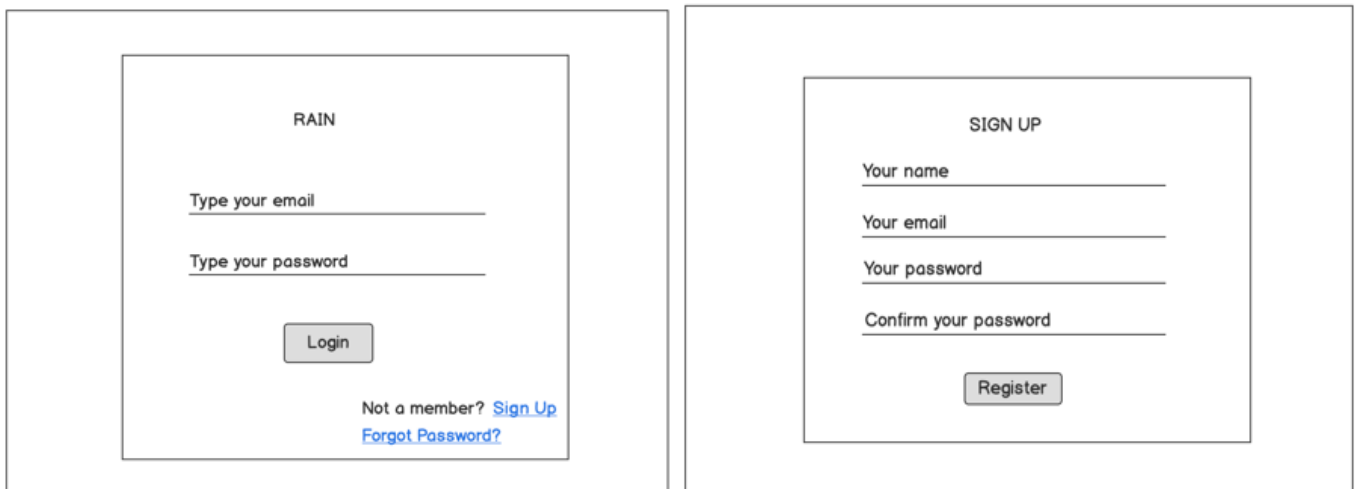


Figure 14 Wireframes of login in and sign up

In the following picture we can see the landing page or dashboard structure which contains maps, table, line charts and different input box like drop down menu and search boxes. The maps components are directly connected to the year slider which makes it possible to make those maps interactive by triggering the data during the moving years and changing the markup layer of each country.

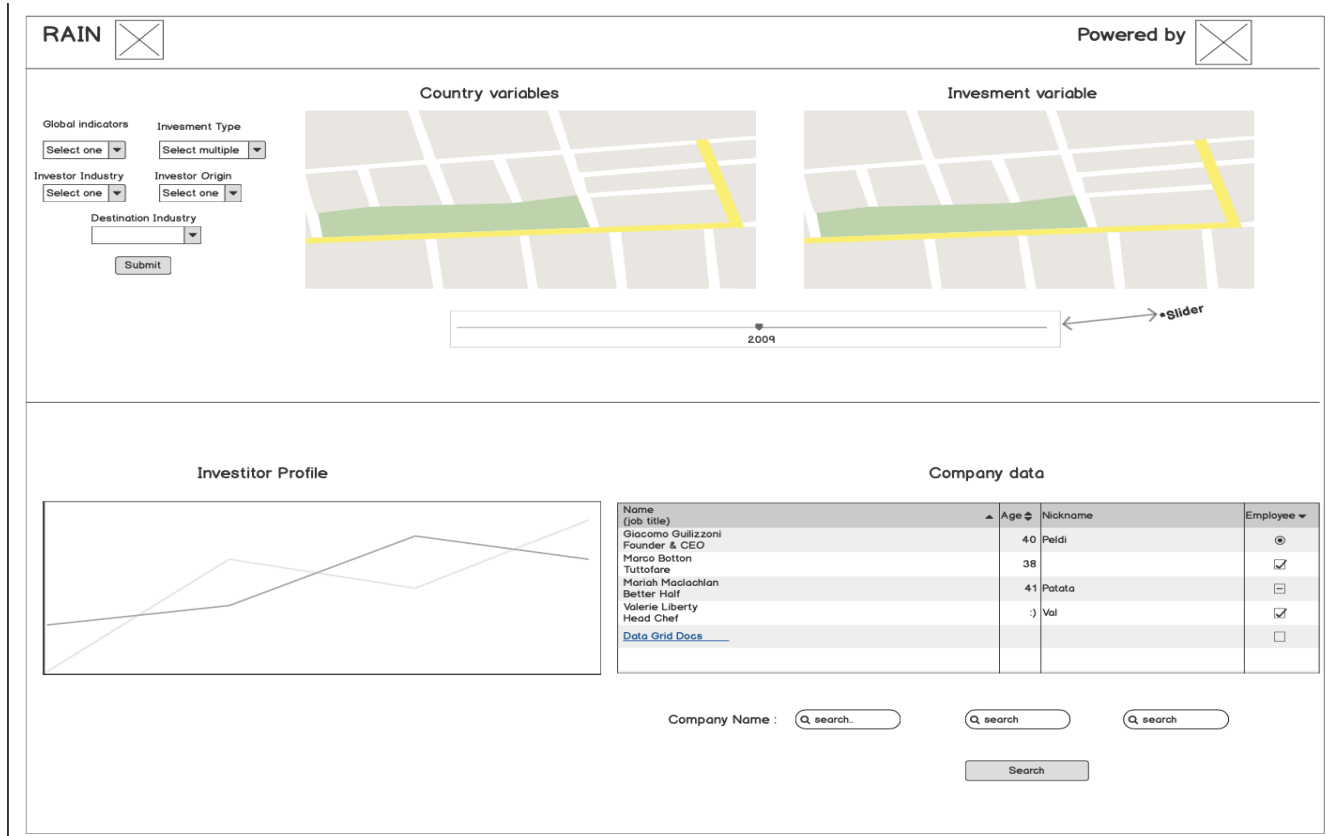


Figure 15 RAIN dashboard wireframe

4.4 Implementation

The implementation phase is going to be structured from designing the layout of the application and their user interface components, to coding the functionality of React functions and their interactivity with json data.

4.4.1 Storybook components

Before positioning the UI components in the web layout, we design it in storybook to see how they will look in different states and what content they will show. This gives us a preview display per each user interface component that we are going to develop in different use cases.

Here we can see one example of the stored components that is previewed for testing and designing before its going to be placed in the landing page. In storybook table we can see the list of all the components that we created to test the functionality of each component before putting those in the real application.



Figure 16 Storybook component

The storybook tool is integrated directly with react framework and whenever is the possibility to add a new potential component is going to be added here (Figure 16) for preview version.

4.4.2 Code implementation of React components

So for developing a visualization dashboard that needs a lot of interactive data, we structured the code into the components. Each component contains logical functions that represent various interactive data.

So the main components of our dashboard application are those as follows:

- GepJsonMap Component
- ClusterMap Component
- LineChart Component
- CompanyTable Component

GeoJSON Map component

Firstly, we gonna describe the most important component in our application called GeoJSON Map. The role of this component is to get the JSON data and visualize them in an interactive map layers.

The GeoJson Map class component is created with the integration of Mapbox-*gl* library which is a mapbox javascript library for creating interactive maps. It can be very easily integrated with react components through a react npm⁷ package called *react-map-gl*

```
import MapGL, { Source, Layer } from "react-map-gl";
```

Here we can see all the features of map positioning, set of year panel and initialization of data property that will be changed from fetching the JSON API data. All of those are stored in state attribute that React has managed to update and pass it through methods within the class, and also to build it in lifecycle methods.

```
1. export default class GeoJsonMap extends Component {  
2.   state = {  
3.     year: 2015,  
4.     data: null,  
5.     hoveredFeature: null,  
6.     viewport: {  
7.       latitude: 40,  
8.       longitude: -100,
```

⁷ Node packet manager – is a packet manager for node runtime environment

```
9.     zoom: 2.5,  
10.    bearing: 0,
```

So in the following code, we can see the *componentDidMount* lifecycle method which stands for calling an API JSON request and mounting the component from the moment that application starts to run.

Here the API JSON is fetched from the internet and it's loaded into the *._loadData* method which it is passed by data parameter in the *setState* react function, where the state of data is updated through a method *updatePercentiles* which will trigger the state of the year and will update the JSON data that will be shown into the interactive Mapbox map.

```
1. componentDidMount() {  
2.     requestJson(  
3.         "https://raw.githubusercontent.com/uber/react-map-gl/master/examples/.data/us-  
income.geojson",  
4.         (error, response) => {  
5.             if (!error) {  
6.                 this._loadData(response);  
7.             }  
8.         }  
9.     );  
10. }  
11.  
12. _loadData = data => {  
13.     this.setState({  
14.         data: updatePercentiles(data, f => f.properties.income[this.state.year])  
15.     });  
16. };  
17.  
18. _updateSettings = (name, value) => {  
19.     console.log(`name:${name}, value:${value}`);  
20.     if (name === "year") {  
21.         this.setState({ year: value });  
22.  
23.         const { data } = this.state;  
24.         if (data) {  
25.             // trigger update  
26.             this.setState({  
27.                 data: updatePercentiles(data, f => f.properties.income[value])  
28.             });  
29.             console.log("data", data);  
30.         }  
31.     }  
32. };
```

At the end of the class, we can see what is the output or return method of the geomap class by mapping all the map gl library features (source and layer) with their suitable values for sizing and map styling.

Also, we can see that there is included another outsider class called control panel which contains the year slider to trigger the interactivity of map color layers with time-series data.

```
1. return (
2.   <>
3.     <div style={{ height: "100%" }}>
4.       <MapGL
5.         {...viewport}
6.         width="100%"
7.         height="100%"
8.         mapStyle="mapbox://styles/mapbox/light-v9"
9.         onViewportChange={this._onViewportChange}
10.        mapboxApiAccessToken={MAPBOX_TOKEN}
11.        onHover={this._onHover}
12.      >
13.        <Source type="geojson" data={data}>
14.          <Layer {...dataLayer} />
15.        </Source>
16.        {console.log("layers", dataLayer)}
17.        {this._renderTooltip()}
18.      </MapGL>
19.
20.      <ControlPanel
21.        containerComponent={this.props.containerComponent}
22.        settings={this.state}
23.        onChange={this._updateSettings}
24.      />
25.    </div>
26.  </>
```

Here we can see control panel class that was included in geojson component, and this class serves as a container for year range slider that we are going to see later in the application.

The class contains an input field with attributes of min, max for range of years and an event for changing the range input type during the passing of year values.

```
1. export default class ControlPanel extends PureComponent {
2.   render() {
3.     const Container = this.props.containerComponent || defaultContainer;
4.     const { settings } = this.props;
5.
6.     return (
7.       <Container>
8.         <div key={"year"} className="input">
```

```

9.         {" "}
10.        <label> Year: {settings.year}</label>
11.        <input
12.            type="range"
13.            value={settings.year}
14.            min={1995}
15.            max={2015}
16.            step={1}
17.            onChange={evt => this.props.onChange("year", evt.target.value)}

```

The most important JavaScript library in the geojson map component is d3 which we have discussed in-depth on the literature review. In the following function (*updatePercentiles*) we can see the implementation of the d3 library, with *d3-array* and *d3-scale*. The first one is used for looping an array of ten items together with the second one which is scaling the elements from the range function and updating as percentiles. All of those combinations from d3 are going to be matched and updated with the structure of JSON data that is fetched from the API, and they are changed accordingly to the initialization of the range input slider from the control panel class.

```

1. import { range } from "d3-array";
2. import { scaleQuantile } from "d3-scale";
3.
4. export function updatePercentiles(featureCollection, accessor) {
5.     const { features } = featureCollection;
6.     const scale = scaleQuantile()
7.         .domain(features.map(accessor))
8.         .range(range(9));
9.     return {
10.         type: "FeatureCollection",
11.         features: features.map(f => {
12.             const value = accessor(f);
13.             const properties = {
14.                 ...f.properties,
15.                 value,
16.                 percentile: scale(value)
17.             };
18.             return { ...f, properties };
19.         })
20.     };
21. }

```

So now we are going to see the screenshot of how the geojson map component looks into the dashboard application. With all the things that we mention before, here we can see map layers that symbolize the density of colors by the higher rank of percentile per each state, and the slider input that has a range from the year 1999 to 2015. Data that are to be rendered are state, median household income, and percentile. Each movement of the slider by year affects the data in the map as it can be seen in tooltip below.

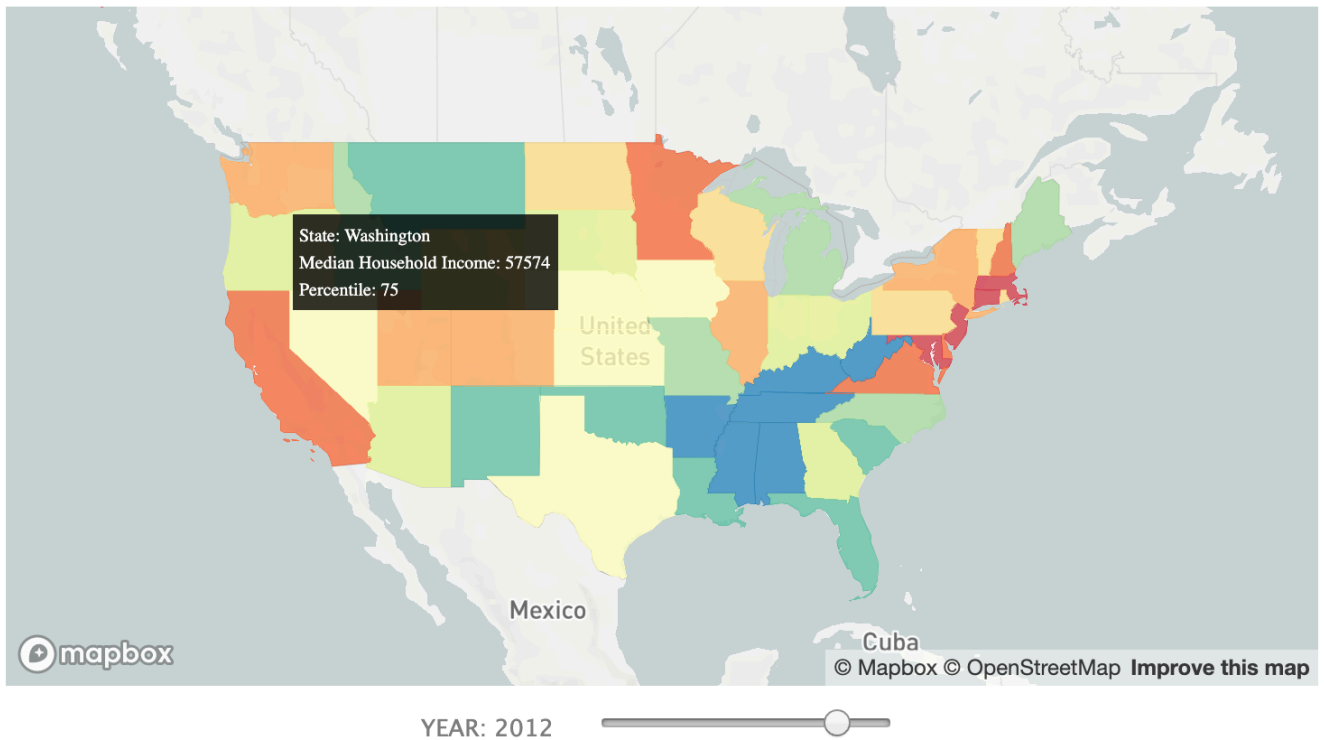


Figure 17 GeoJson Map

ClusterMap Component

As we talked about the first geojson map component, implementation of this component is almost the same with *mapbox-gl* library but with some differences in data fetching and structuring of geographical coordinates.

So, for code overview here we are going to describe briefly the most important functions of this class component.

The only functionality of the `map` is an `onClick` event function that allows to zoom and transfer `cluster layer` by click upon it. This happens by getting cluster id and index of features to match it with longitude and latitude of country positioning.

```
1. _onClick = event => {
2.   const feature = event.features[0];
3.   const clusterId = feature.properties.cluster_id;
4.
5.   const mapboxSource = this._sourceRef.current.getSource();
6.
7.   mapboxSource.getClusterExpansionZoom(clusterId, (err, zoom) => {
8.     if (err) {
9.       return;
10.    }
11.
12.    this._onViewportChange({
13.      ...this.state.viewport,
14.      longitude: feature.geometry.coordinates[0],
15.      latitude: feature.geometry.coordinates[1],
16.      zoom,
17.      transitionDuration: 500
18.    });
19.  });
20.  };
```

The return method of this component is the same with the previous one, just the difference is in the data are fetched directly from the source method method of `mapbox-gl`. Here as well we can see specification for defining the cluster options like the maximum zoom and radius of the cluster.

```
1. return (
2.
3.   .....
4.
5.   <Source
6.     type="geojson"
7.     data="https://docs.mapbox.com/mapbox-gl-js/assets/earthquakes.geojson"
8.     cluster={true}
9.     clusterMaxZoom={14}
10.    clusterRadius={50}
11.    ref={this._sourceRef}
12.  >
13.
14. </Source>
```

15. </

So, here is the snapshot of cluster map implementation into the dashboard, as we can see the states that are provided with data, are highlighted with clusters, each click into the cluster will zoom the map and will spilt the clusters in small parts depending of which cities and regions are affected by given disperse data.

This data spreading is going to be triggered by the input year slider in the first geojson component, which will indicate how the data changed during the ten-year period.

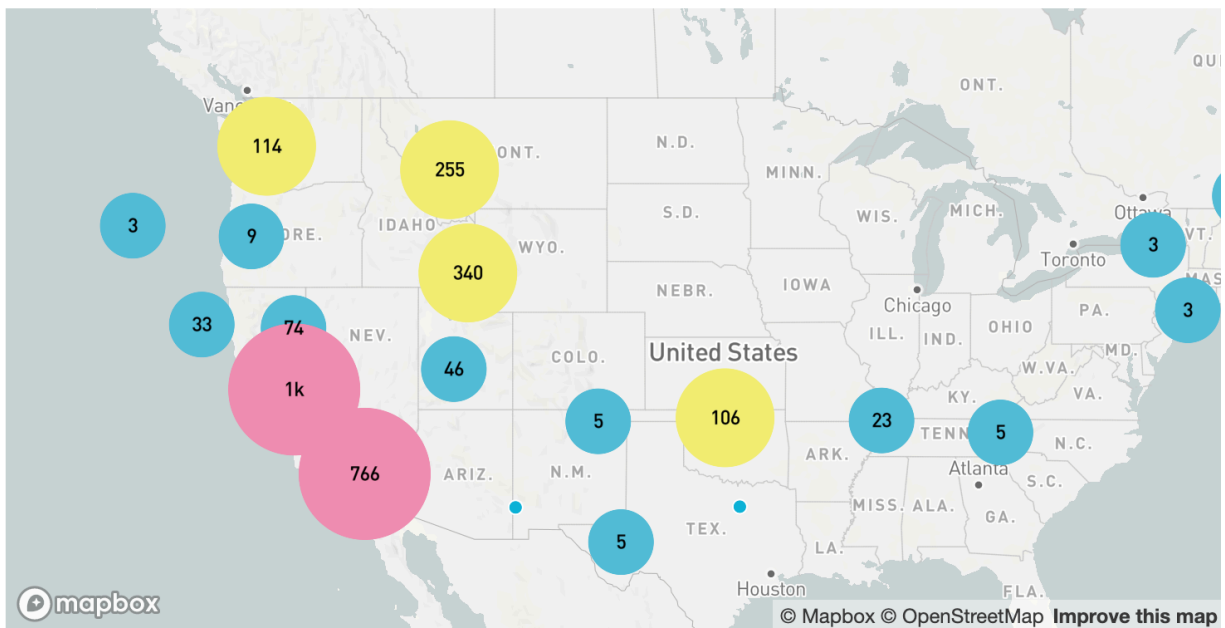


Figure 18 Cluster Map

Line-chart Component

Line chart server as visualization for the information in the data tables component. So, whenever a column of the data table is clicked, the line chart will visualize the selected data into the lines with labels and categories name.

Line charts are built with some elements of d3 and with an improvised javascript library for react called rechart. Rechart contains a separate react component for building the charts like a tooltip, grid and line items.

As we can see in the following picture, the line chart component has curved lines and a label that describe the information of data on the particular point, and two data sources for displaying the information simultaneously, depending on how many columns are going to be selected.

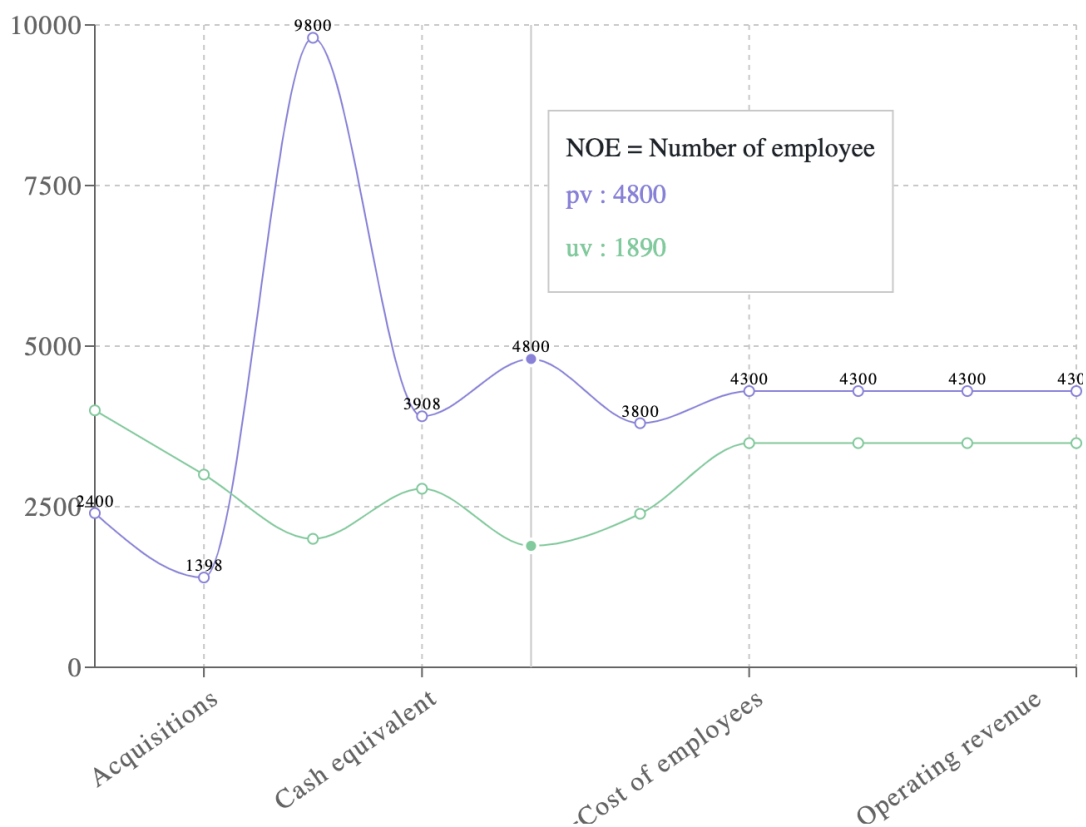


Figure 19 Line Chart

CompanyTable Component

```
import BootstrapTable from "react-bootstrap-table-next";
```

The table is developed with a customized bootstrap library for react named *react-bootstrap-table-next* and is imported as react wrapper component. So it's the use of a bootstrap framework that is reconstructed with a small component to match the architecture of react. It takes data as an object of arrays and each named property of an object is looped with each column that contains an id key called *dataField* with the same value as property in an object array. For displaying the label name of the column is used the keyword *text* and for sorting the column the Boolean property *sort*

```
1. const columns = [  
2.   {  
3.     dataField: "CompanyName",  
4.     text: "Company Name",  
5.     sort: true  
6.   },  
7.   {  
8.     dataField: "GUOName",  
9.     text: "GUO - Name",  
10.    sort: true  
11.  },  
12.  {  
13.    dataField: "BvDIDnumber",  
14.    text: "BvD ID number",  
15.    sort: true  
16.  } . ...
```

The return method of the table are the properties that initialize the data, columns, and the other styling features for hovering and making the table with stripes. Nevertheless, an important feature is *paginationFactory*, which is imported as a new bundle to be integrated automatically in bootstrap table and server as paginator of the table. In the end, the component is exported to be used on the main page with other components as a single page application.

```
import paginationFactory from "react-bootstrap-table2-paginator";
```

```
1. return (
```

```

2.     <>
3.     <BootstrapTable
4.         className="bTable"
5.         keyField="id"
6.         data={CompanyData}
7.         columns={columns}
8.         pagination={paginationFactory()}
9.         striped
10.        hover
11.        condensed
12.
13.    />
14. </>
15. );
16. }
17. }
18.
19. export default CompanyTable;

```

Here is the snapshot of table component with all the rendered data and stripe styled columns, also we can see the paginator factory bundle that is automatically styled on the bottom of the table as a filter for the columns with selected numbers per page, and < previous – next > buttons

Company Name	GUO - Name	BvD ID number	Country	Number of deals	Number of projects	Major sectors
WALMART INC.	WALMART INC.	US710415188	United States of America	17	113	Wholesale & retail trade
JOINT-STOCK COMPANY VTB CAPITAL	VTB BANK (PUBLIC JOINT-STOCK COMPANY)	RU94102445	Russian Federation	8	1	Banks
SAUDI ARAMCO COMPANY	SAUDI ARAMCO COMPANY	SA0000026859	Saudi Arabia	14	22	Other services

10
▼

- 10
- 25
- 30
- 50

<
1
2
3
>

Figure 20 Company table with pagination

4.4.3 Demonstration of RAIN dashboard application

After the code review implementation of each component, now is time to demonstrate the RAIN dashboard step by step as a single page application.

The first step that the user does when tries to access the RAIN dashboard is to register in the sign-in page, after it is registered, he can log in RAIN login page with credentials that he gets from the confirmation email. If he potentially forgot the password, he can submit the via link text below the login button.

The image shows two side-by-side form panels. The left panel is titled "Sign up" and contains four input fields: "Your name", "Your email", "Confirm your email", and "Your password". A teal "REGISTER" button is positioned at the bottom center. The right panel is titled "RAIN" and contains two input fields: "Type your email" and "Type your password". A blue "LOGIN" button is centered below the password field. At the bottom right of the right panel, there are two links: "Not a member? Sign Up" and "Forgot Password?".

Figure 21 Sign up & Login form

After the user is successfully logged in, he can see the landing page of the RAIN dashboard with all the named components for filtering and searching the data. Maps are served as indicator points of data spreading throughout the states of the USA. They are two types of maps one is for the origin of investors and the other is for places that investors have made their investments. The main filtering option of maps is the slider component which is used as a sequential year filter for showing how the data change during the years in different states of

America. This filtering feature is combined with a hover tooltip that describes in more detail the information about each state during the transition of the years in the USA.

In the bottom part of the dashboard, we can see a data table that displays different companies across all the world that have made various investments in the united states of America. The data table also describes the numbers of projects and deals that each company has made in different major sectors like banks, retail trade, machinery, etc. Line chart visualizes the company data with line curves and detailed information in a tooltip and also it shows the main categories in labels across the line. When a column is clicked into the company table cell it transformed immediately the data into the line chart visualization form with line curves that goes according to the ranking labels numbers.

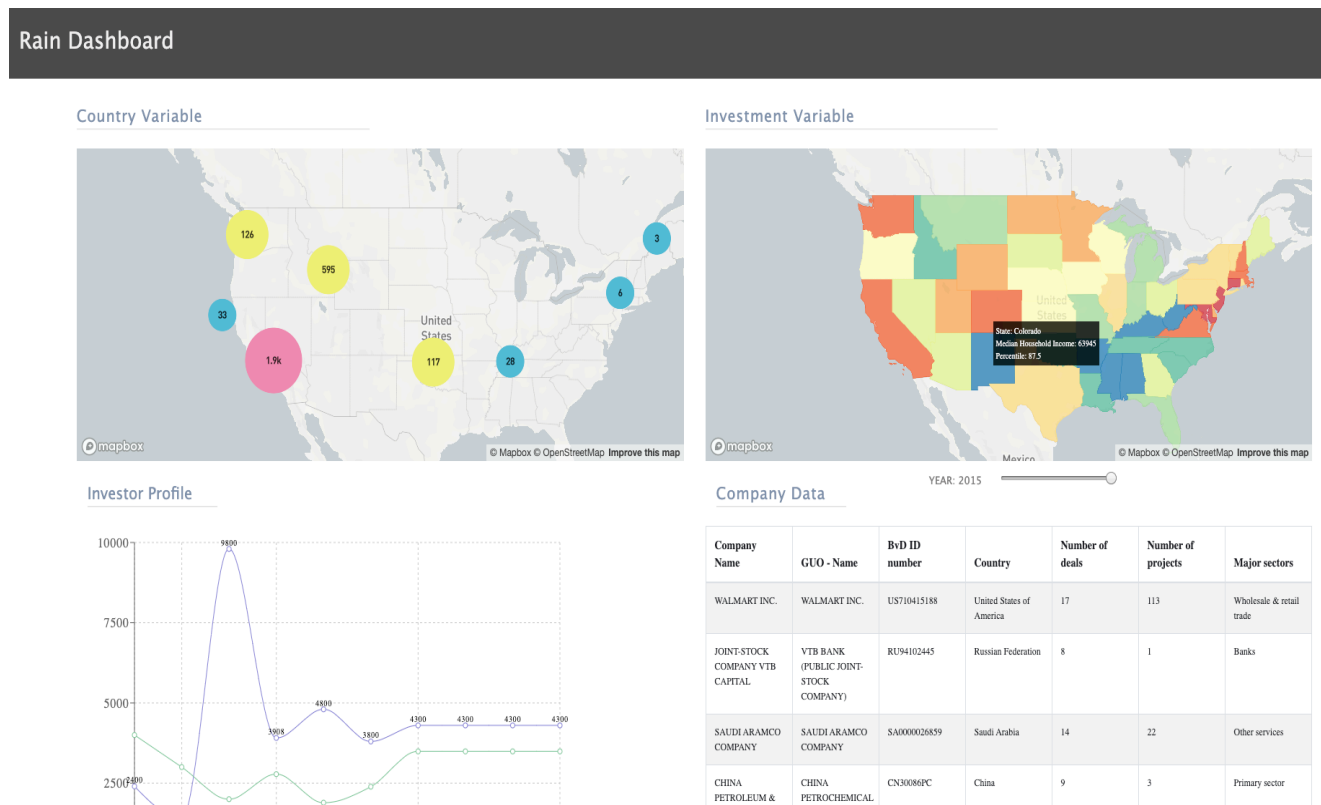


Figure 22 RAIN Dashboard

4.5 Testing

For testing our dashboard application we are going to use JavaScript testing framework called Jest that is developed by Facebook and its most suitable for react applications.

We are going to use unit testing to check how many uses cases are passing successfully to the most important components of the dashboard.

The first test will be to check if the company data are rendered properly, so we wrote a test case to look for the first index of the data and to match it with the expected one.

Firstly we created a variable called *CompanyData* which server to import the path where the data were stored, and after that, we initialize the *CompanyData* variable with index zero by using the jest matcher *expect*. After that, we use another matcher - *toEqual* to match the first object in an array of company data for the first index of company data array of objects.

```
1. const CompanyData = require('../mockData/CompanyData');
2.
3. test('Checking the first index of company data ', () =>{
4.
5.     expect(CompanyData[0]).toEqual({  CompanyName: "WALMART INC.",
6.         GUOName: "WALMART INC.",
7.         BvDIDnumber: "US710415188",
8.         Country: "United States of America",
9.         NumberOfDeals: 17,
10.        NumberOfProjects: 113,
11.        MajorSectors: "Wholesale & retail trade"})
12.
13. })
```


The results of the test were successful with one test case in total, and the time of running was less than one second .

```
> rain_v2@0.1.0 test /Users/i513931/Documents/RainDashboard_v2/rain_v2
> jest "CompanyTbl.test.js"

PASS src/test/CompanyTbl.test.js
  ✓ Checking the first index of company data (3ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:  0 total
Time:        0.934s, estimated 1s
Ran all test suites matching /CompanyTbl.test.js/i.
```

We tested the other source of data that appears in the line chart visualization. This data is for investor profile that is related with company data. The test case was kind of similar to the previous one.

The test case is checking if the first object of an array is matched with their couple given data.

```
1. const data = require('../mockData/ChartData')
2.
3.
4. test('Checking the first element of investor profile data', () =>{
5.
6.     expect(data[0]).toEqual({ name: "Investments",
7.       uv: 4000,
8.       pv: 2400,
9.       amt: 2400})
10.
11. })
```

As we can see test scenario has passed successfully without any error.

```
PASS src/test/LineChartData.test.js
  ✓ Checking the first element of line chart data

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:  0 total
Time:        0.922s, estimated 1s
Ran all test suites matching /LineChartData.test.js/
```

If we try to select the second object from the array, we can see that the test will fail because it will not match with the given comparing data.

Here we can see the differences of the expected and received data, the first ones with green color tell us the given comparing data which represents the first object, and the second ones are the input data that comes from the second object.

The jest is commenting on this test case with “deep equality” which says to us that equality of two objects is very similar except for different values.

```
× Checking the first element of investor profile data (9ms)
● Checking the first element of investor profile data
  expect(received).toEqual(expected) // deep equality

  - Expected   - 4
  + Received   + 4

  Object {
    - "amt": 2400,
    - "name": "Investments",
    - "pv": 2400,
    - "uv": 4000,
    + "amt": 2210,
    + "name": "Acquisitions",
    + "pv": 1398,
    + "uv": 3000,
  }

   4 | test('Checking the first element of investor profile data
     |
   > 6 |   expect(data[1]).toEqual({ name: "Investments",
     |                         ^
     |   uv: 4000,
     |   pv: 2400,
     |   amt: 2400})
     |
     |   at Object.<anonymous>.test (src/test/LineChartData.test.js:6:22)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:  0 total
Time:        2.046s
Ran all test suites matching /LineChartData.test.js/i.
npm ERR! Test failed.  See above for more details.
```

5 Result and Discussion

5.1 React and single page dashboard

From consulting the literature review of react js we can see the advantages that react has in the aspect of processing data efficiently with one-way data binding, virtual dom, and server-side scripting. All of those contribute very highly In rendering the web dashboard very smoothly for single page application. The asynchronous approach that React adopts is very crucial for making the dashboard very responsive to different requests that get from data servers, especially with javascript promises that make handling of API very efficiently.

Virtual DOM helps the react very much to minimize the use of the browser which leads to the better layout page performance. (Modern and Responsive Mobile-enabled Web Applications, 2017).

All of those server-side aspects react handles it very easily with the integration of node js runtime environment which is very suitable for combining the client-side with server-side via using the same template to render into the browser (2015)

Another aspect that combines very well react with the architecture of SPA is the integration of generic libraries like Mapbox gl that makes the map components of the dashboard very interactive by showing the data asynchronously in forms of map layers and circles across the different countries of the world.

Meanwhile, the most fundamental aspect that helped the pure implementation of a single page dashboard was the introduction of the AJAX web development technique which made it very easy to load only some part of web application without needing to render the whole page.

Also, this helped the bandwidth of the server to be more lightweight and faster in processing the different requests that came from the user in real-time. Having into the consideration that a single page dashboard contains a lot of components that need to be reloaded in a short period, the usage of AJAX technique is essential for re-rendering the components in an efficient way. (Paul Colton, 2011)

We can conclude that with all the features that react possess like virtual dom and server-side scripting together with the integration of different data-driven s javascript libraries, it

completes the requirement that single page dashboard application demand to perform in high availability way. Especially in terms of data flowing through the components in the dashboard which are very generic and demanding to be updated in a daily basis.

5.2 Comparison of React and other JavaScript frameworks

Seeing the benchmark(Figure 5) comparisons and differences of react features that we did in previous chapters, we can conclude that react was more dominant in most the table operations fields like replacing, swapping, clearing and appending rows to a large table. All of those were measured by duration in milliseconds per each operation. Another thing that differentiates react in the aspect of community support is that from GitHub statistics (Figure 4) we can see the popularity that React has in aspects of contributing to the framework and helping the community with a big support group.

The fact also we can see in one JavaScript survey website (Raphaël, 2019) that ranks yearly each js framework by the aspect of awareness, interest and ratio satisfaction.

For example, in this graph, we can see the level of satisfaction that users had for javascript frameworks during the years, and as we can see react was leading from 2016 with an expectation in 2018 where vue was rising but on the same level as react.

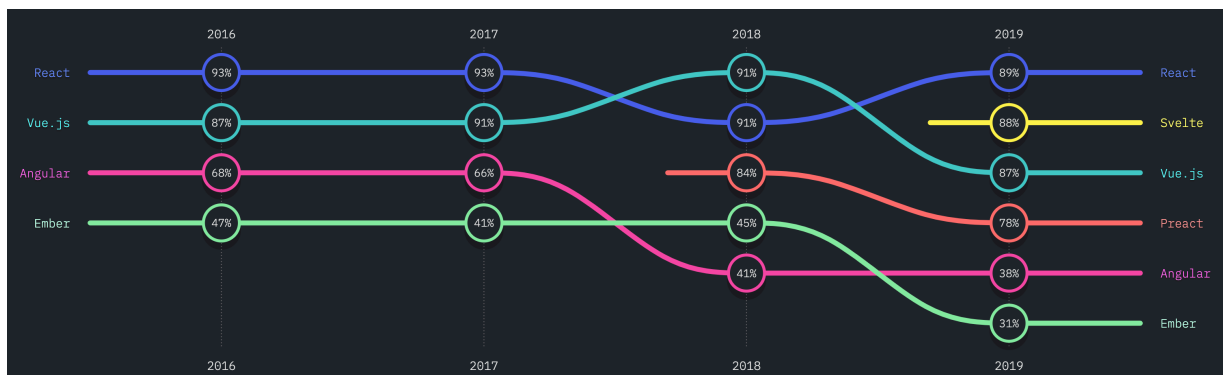
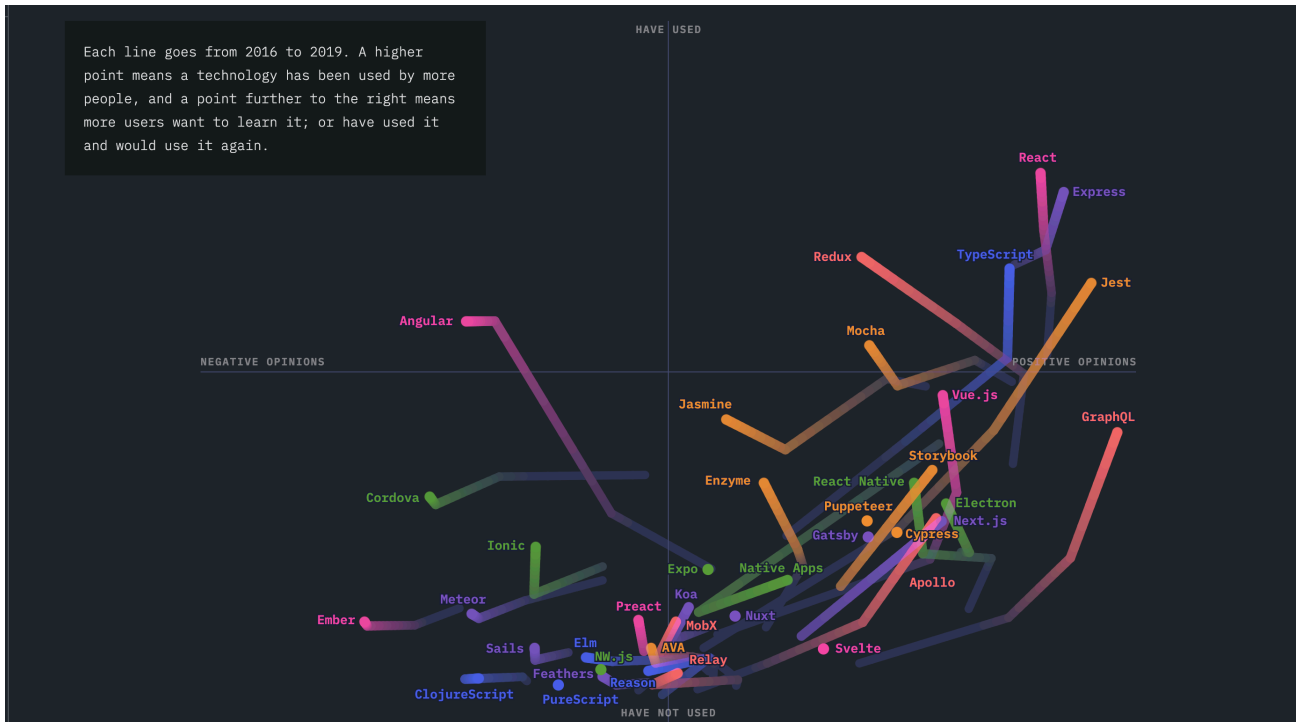


Figure 23 Rankings of front end frameworks (Raphaël, 2019)

Similar to the previous graph here we can see the full ecosystem of JavaScript from 2016 to 2019 in aspect of usage and reviews.



(Wattenberger, 2019)Figure 24 Changes of JavaScript ecosystem during the years

As we mentioned earlier in the literature review what differentiates react from other JavaScript frameworks is the use of one-way data binding which makes the flow of the data more structured and easier to control. The passing of the data starts from one global state to another children's components, so the updating of the user interface happens only if the global state is changed. Meanwhile, the other JavaScript frameworks, in this case, Angular uses two-way data binding which means that the change of the data is going to be affected by global state and user interface at the same time.

6 Conclusion

The main objective of the thesis was to test and evaluate the performance of the React framework into the advanced visualization dashboard as a single page application, and to make a formal comparison between the most famous JavaScript frameworks in terms of performance.

The first partial objective was to develop a web dashboard as a single page application with a react framework that will display various economic and geographic data into different react components like interactive maps, tables and line charts using different libraries of JavaScript. The second partial objective was to test the dashboard performance and to compare it with other JavaScript frameworks. This was done by a benchmark script that analyzed the performance of the top three JavaScript frameworks in terms of table operations, data entry and memory usage.

In the end, we can conclude that the whole research, implementation, and evaluation was done almost at the required level with some limitations in the aspect of the development of the dashboard application which was more in testing phase using the mock data and running only in localhost. Also, we developed the application only in one JavaScript framework which left us a bit uncomplete for testing the performance of three identical dashboard applications in different JavaScript frameworks.

So, this leaves us with much work to do in the future, especially in deploying and maintaining the application in a real server and using the data from the hosting database server.

Nevertheless, further research of JavaScript's impact in developing the web dashboard is very needed having into consideration the rapid evolvement that web technologies are having in these days.

7 Bibliography

Trends in Web Based Cross. **Smeets, Ruben and Aerts, Kris. 2016.** Issue. 6, s.l. : IJCSMC, 2016, International Journal of Computer Science and Mobile Computing, Vols. Vol. 5,, pp. 190-199. ISSN 2320-088X.

Niclas Hansson, Tomas Vidhall. 2016. Effects on performance and usability for cross-platform application development using React Native. <http://www.diva-portal.org/>. [Online] June 16, 2016. [Cited: February Saturday, 2019.] <http://www.diva-portal.org/smash/get/diva2:946127/FULLTEXT01.pdf>. LIU-IDA/LITH-EX-A-16/043-SE.

Lebensold, Jonathan. 2018. *React Native Cookbook: Bringing the Web to Native Platforms.* Sebastopol : O'Reilly Media, Inc., 2018. 1491993790, 9781491993798.

<https://www.czso.cz/>. [Online]

Chalaris, Angelos. 2018. Javascript Framework Comparison with Examples (React, Vue & Hyperapp). *www.hackernoon.com*. [Online] July 2018. <https://hackernoon.com/javascript-framework-comparison-with-examples-react-vue-hyperapp-97f064fb468d>.

Chand, Swatee. 2019. What Is React? – Unveil The Magic Of Interactive UI With React. <https://www.edureka.co>. [Online] March 2019. <https://www.edureka.co/blog/what-is-react/>.

Johnny. 2017. The Benefits of Using React. <https://www.sourcetoad.com>. [Online] January 24, 2017. <https://www.sourcetoad.com/app-development/the-benefits-of-using-react/>.

dhtmlx. 2019. JavaScript Trends in 2020. *codebrus*. [Online] December 23, 2019. <https://codeburst.io/javascript-trends-in-2020-b194bebc5ef8>.

Moraga, Paula. 2019. *Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny.* Boca Raton, Florida : CRC Press Taylor & Friends Group, 2019.

2019. Developer Survey Results 2019. *stackoverflow*. [Online] 2019. <https://insights.stackoverflow.com/survey/2019>.

Subramanian, Vasan. 2017. *Pro MERN Stack Full Stack Web App Development with Mongo, Express, React, and Node.* Bangalore, Karnataka, India : Apress, 2017.

Box, Juice. 2015. A Guide to Creating Dashboards People Love to Use. <https://www.juiceanalytics.com/juicebox>. [Online] May 2015. https://static1.squarespace.com/static/52f42657e4b0b3416ff6b831/t/55b9117ae4b060a0d84fef15/1438191994754/Dashboards_People_Love_To_Use_Whitepaper_v2.pdf.

Gackenheimer, Cory. 2015. *Introduction to React.* New York : apress, 2015.

2014. Draft: JSX Specification. *JSX*. [Online] Facebook, Inc., 2014. <https://facebook.github.io/jsx/>.

- Daityari, Shaumik. 2020.** Angular vs React vs Vue: Which Framework to Choose in 2020 . *codeinwp*. [Online] February 18, 2020. [Cited: February 23, 2020.] <https://www.codeinwp.com/blog/angular-vs-vue-vs-react/>.
- Mardan, Azat. 2014.** *Pro Express.js*. s.l. : Apress, 2014. 978-1-4842-0037-7.
- Meeks, Elijah. 2018.** *D3.js in action : data visualization with JavaScript*. Manning : Shelter Island , 2018. 978-1-61729-448-8.
- Bostock., Mike. 2019.** Data-Driven Documents. *d3js*. [Online] 2019. <https://d3js.org/>.
- Gackenheim, Cory. 2015.** *Introduction to React*. New York : Apress, 2015. 978-1-4842-1245-5.
- Wexler, Steve , Shaffer, Jeffrey and Cogtave, Andy. 2017.** *The Big Book of Dashboards : Visualizing Your Data Using Real-World Business Scenarios*. New Jersey : John Wiley & Sons, Inc., Hoboken, New Jersey, 2017.
- Overview of Web Development Life cycle in Software Engineering.* **Sarkar, Ashim. 2018.** 6, Ranchi Jharkhand India : International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2018, Vol. 3. 2456-3307.
- Milligan, Joshua N. 2015.** *Learning Tableau Leverage the power of Tableau 9.0 to design rich data visualizations and build fully interactive dashboards*. BIRMINGHAM - MUMBAI : Packt Publishing, 2015. 978-1-78439-116-4.
- Aspin, Adam. 2016.** *Pro Power BI Desktop*. Apress : Berkeley, CA , 2016. 978-1-4842-1805-1 .
- Alexander, Michael and Walkenbach, John. 2013.** *Excel Dashboards and Reports*. New Jersey : John Wiley & Sons, 2013. 978-1-118-49042-6.
- Paul Colton, Uri Sarid, Kevin Edward Lindsey. 2011.** *DASHBOARD FOR ON-THE-FLY AJAX MONITORING* . US 7,958,232 B1 United States of America, June 7, 2011.
- Creating and Using Personas in Software Development: Experiences from Practice.* **Nielsen, Jane Billestrup Jan Stage Anders Bruun Lene Nielsen Kira S. 2014.** Berlin, Heidelberg : IFIP International Federation for Information Processing, 2014. 978-3-662-44811-3.
- Friedman, Vitaly. 2012.** *User Experience Practical Techniques* . Freiburg, Germany : Smashing Media GmbH, 2012. 978-3-943075-25-0.
- Adiseshiah, Emily Grace. 2017.** Introducing Justinmind's new UI kit: Charts. *justinmind*. [Online] June 19, 2017. [Cited: March 16, 2020.] <https://www.justinmind.com/blog/introducing-justinminds-new-ui-kit-charts/>.
- Sole, Alessandro Del. 2018.** *Visual Studio Code Distilled : Evolved Code Editing for Windows, MacOS, and Linux*. s.l. : Apress L. P., 2018. 9781484242230.
- Faranello, Scot. 2012.** *Balsamiq Wireframes Quickstart Guide*. Birmingham : PACKT Publishing, 2012. 978-1-84969-352-3.

Low vs. High-Fidelity Prototyping Debate. **Rudd, Jim and Stern, Ken and Isensee, Scott. 1996.** s.l. : Association for Computing Machinery, 1996, Vol. 3. 1072-5520.

Vegh, Aaron. 2010. Wireframe Basics. *Web Development with the Mac.* s.l. : John Wiley & Sons, Incorporated, 2010.

Modern and Responsive Mobile-enabled Web Applications. **Shahzad, Farrukh. 2017.** Houston : Elsevier B.V, 2017, Vol. 110.

Wattenberger, Amelia. 2019. *Fullstack D3 and Data Visualization: Build beautiful data visualizations with D3.* s.l. : Fullstack.io , 2019. 978-0991344659.

Raphaël, Sacha and. 2019. Front End Frameworks. *State of JS.* [Online] 2019. [Cited: March 29, 2020.] <https://2019.stateofjs.com/>.

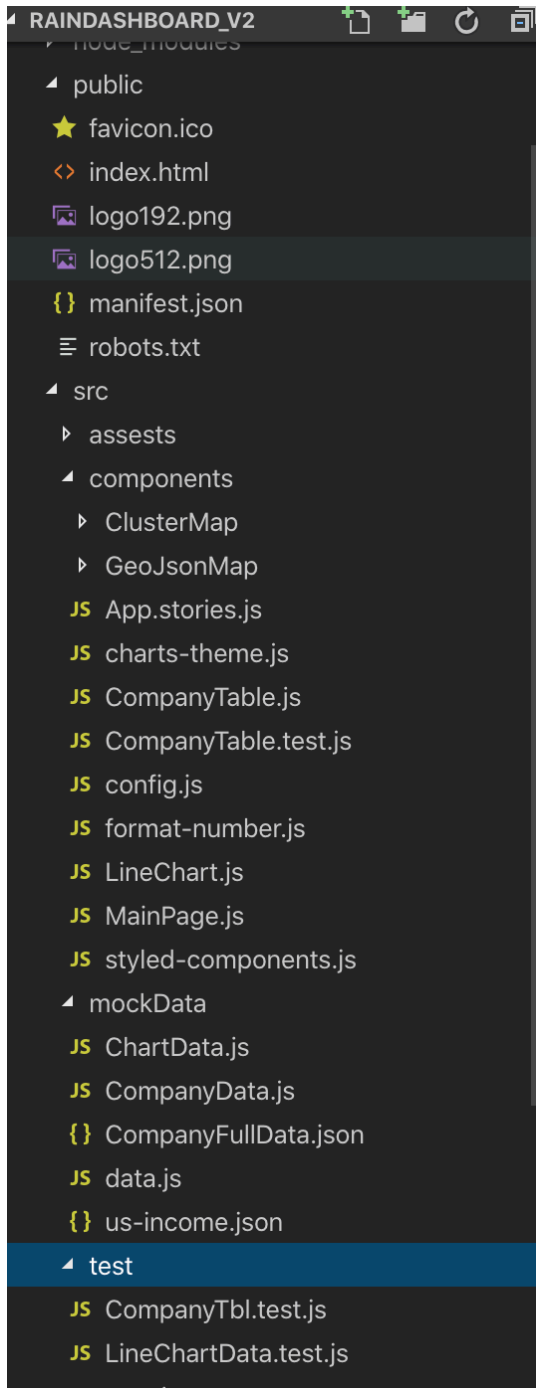
2015. Why we use React for building high performance single page applications. *Graph.* [Online] December 17, 2015. [Cited: March 29, 2020.] <https://www.graph.uk/blog/why-we-use-reactjs-for-building-single-page-applications>.

Gomaa, Hassan. 2011. Use cases. *Software Modeling and Design* . Virginia : Cambridge University press, 2011.

Connect. [Online] <https://github.com/senchalabs/connect#readme>.

8 Appendix

Application structure of rain dashboard



Mock data of company table

```
1. const CompanyData =[
2.   {
3.     CompanyName: "WALMART INC.",
4.     GUOName: "WALMART INC.",
5.     BvDIDnumber: "US710415188",
6.     Country: "United States of America",
7.     NumberOfDeals: 17,
8.     NumberOfProjects: 113,
9.     MajorSectors: "Wholesale & retail trade"
10.  },
11.  {
12.    CompanyName: "JOINT-STOCK COMPANY VTB CAPITAL",
13.    GUOName: "VTB BANK (PUBLIC JOINT-STOCK COMPANY)",
14.    BvDIDnumber: "RU94102445",
15.    Country: "Russian Federation",
16.    NumberOfDeals: 8,
17.    NumberOfProjects: 1,
18.    MajorSectors: "Banks"
19.  },
20.  {
21.    CompanyName: "SAUDI ARAMCO COMPANY",
22.    GUOName: "SAUDI ARAMCO COMPANY",
23.    BvDIDnumber: "SA000026859",
24.    Country: "Saudi Arabia",
25.    NumberOfDeals: 14,
26.    NumberOfProjects: 22,
27.    MajorSectors: "Other services"
28.  },
29.
30.  {
31.    CompanyName: "CHINA PETROLEUM & CHEMICAL CORPORATION",
32.    GUOName: "CHINA PETROCHEMICAL CORPORATION",
33.    BvDIDnumber: "CN30086PC",
34.    Country: "China",
35.    NumberOfDeals: 9,
36.    NumberOfProjects: 3,
37.    MajorSectors: "Primary sector"
38.  },
39.  {
40.    CompanyName: "ROYAL DUTCH SHELL PLC",
41.    GUOName: "ROYAL DUTCH SHELL PLC",
42.    BvDIDnumber: "GB04366849",
43.    Country: "United Kingdom",
44.    NumberOfDeals: 38,
45.    NumberOfProjects: 58,
46.    MajorSectors: "Primary sector"
47.  },
48.  {
49.    CompanyName: "VOLKSWAGEN AG",
50.    GUOName: "PORSCHE AUTOMOBIL HOLDING SE",
51.    BvDIDnumber: "DE2070000543",
52.    Country: "Germany",
53.    NumberOfDeals: 12,
54.    NumberOfProjects: 49,
55.    MajorSectors: "Machinery, equipment, furniture, recycling"
```

```

56.   },
57.   {
58.     CompanyName: "PETROCHINA COMPANY LIMITED",
59.     GUOName: "CHINA NATIONAL PETROLEUM CORPORATION",
60.     BvDIDnumber: "CN30081PC",
61.     Country: "China",
62.     NumberOfDeals: 5,
63.     NumberOfProjects: 4,
64.     MajorSectors: "Primary sector"
65.   },
66.   {
67.     CompanyName: "TOYOTA MOTOR CORPORATION",
68.     GUOName: "TOYOTA MOTOR CORPORATION",
69.     BvDIDnumber: "JP1180301018771",
70.     Country: "Japan",
71.     NumberOfDeals: 6,
72.     NumberOfProjects: 127,
73.     MajorSectors: "Machinery, equipment, furniture, recycling"
74.   }

```

Mock data of line chart

```

1.  const data = [
2.    {
3.      name: "Investments",
4.      uv: 4000,
5.      pv: 2400,
6.      amt: 2400
7.    },
8.    {
9.      name: "Acquisitions",
10.     uv: 3000,
11.     pv: 1398,
12.     amt: 2210
13.   },
14.   {
15.     name: "PL(Before Tax)",
16.     uv: 2000,
17.     pv: 9800,
18.     amt: 2290
19.   },
20.   {
21.     name: "Cash equivalent",
22.     uv: 2780,
23.     pv: 3908,
24.     amt: 2000
25.   },
26.   {
27.     name: "NOE = Number of employee",
28.     uv: 1890,
29.     pv: 4800,
30.     amt: 2181
31.   },
32.   {
33.     name: "ER = Export Revenue",

```

```

34.     uv: 2390,
35.     pv: 3800,
36.     amt: 2500
37.   },
38.   {
39.     name: "COE =Cost of employees",
40.     uv: 3490,
41.     pv: 4300,
42.     amt: 2100
43.   },
44.   {
45.     name: "R & D expenses = RDE",
46.     uv: 3490,
47.     pv: 4300,
48.     amt: 2100
49.   },
50.   {
51.     name: "PM = profit margin",
52.     uv: 3490,
53.     pv: 4300,
54.     amt: 2100
55.   },
56.   {
57.     name: "Operating revenue",
58.     uv: 3490,
59.     pv: 4300,
60.     amt: 2100
61.   }
62. ];
63. module.exports = data

```

Main page of dashboard

```

1. render() {
2.   return (
3.     <Container>
4.       { /* static navbar - top */ }
5.       <Nav className="navbar navbar-expand-lg fixed-top is-white is-dark-text">
6.         <Container className="navbar-brand h1 mb-0 text-large font-large">
7.           <h3> Rain Dashboard </h3>
8.         </Container>
9.         <Container className="navbar-nav ml-auto">
10.          <Container className="user-detail-section">
11.            { /* <img src={UserImg} className="rounded-circle" alt="user" /> */ }
12.          </Container>
13.        </Container>
14.      <Container className="container-fluid pr-5 pl-5 pt-5 pb-5">
15.        { /* row 1 - revenue */ }
16.        <Container className="row">
17.          <Container className="col-lg-3 col-sm-6 is-light-text mb-4">
18.            <Container className="is-dark-text-light letter-spacing text-large">
19.              Country Variable
20.            </Container>

```

```

21.         </Container>
22.         <Container className="col-lg-3 col-sm-6 is-light-text mb-
23.         4"></Container>
24.         <Container className="col-lg-3 col-sm-6 is-light-text mb-4">
25.             <Container className="is-dark-text-light letter-spacing text-large">
26.                 Investment Variable
27.             </Container>
28.         </Container>
29. <Container className="row" style={{ minHeight: "400px" }}>
30.     <Container className="col-md-6 mb-4">
31.         <Container className="chart-container large full-height">
32.             <ClusterMap />
33.         </Container>
34.     </Container>
35.     <Container className="col-md-6 mb-4">
36.         <Container className="chart-container large full-height">
37.             <div id="map">
38.                 <GeoApp />
39.             </div>
40.     </Container>
41. <Container className="row" style={{ minHeight: "400px" }}>
42.     <Container className="col-md-6 mb-4">
43.         <Container className="col-lg-3 col-sm-6 is-light-text mb-4">
44.             <Container className="is-dark-text-light text-large">
45.                 Investor Profile
46.             </Container>
47.         </Container>
48.     <Container className="chart-container large full-height">
49.         <ReChart />
50. </Container>
51. <Container className="col-md-6 mb-4">
52.     <Container className="col-lg-3 col-sm-6 is-light-text mb-4">
53.         <Container className="is-dark-text-light letter-spacing text-
54.         large">
55.             Company Data
56.         </Container>
57.     </Container>
58.     <Container className="chart-container large full-height">
59.         <CompanyTable />
60.     </Container>
61. </Container>
62. </Container>
63. </Container>
64.     { /* content area end */ }
65. </Container>
66. );
67. }
68. }
69.
70. export default MainPage;

```

Control panel of GeoJson map

```
1. import React, { PureComponent } from "react";
2.
3. const defaultContainer = ({ children }) => (
4.   <div className="control-panel">{children}</div>
5. );
6.
7. export default class ControlPanel extends PureComponent {
8.   render() {
9.     const Container = this.props.containerComponent || defaultContainer;
10.    const { settings } = this.props;
11.
12.    return (
13.      <Container>
14.        <div key={"year"} className="input">
15.          {" "}
16.          <label> Year: {settings.year}</label>
17.          <input
18.            type="range"
19.            value={settings.year}
20.            min={1995}
21.            max={2015}
22.            step={1}
23.            onChange={evt => this.props.onChange("year", evt.target.value)}
24.          />
25.        </div>
26.      </Container>
27.    );
28.  }
29. }
```