Data visualization for the modern web
A look into tools and techniques for visualizing data in Angular 5 applications

Tobias Almroth

20 September 2018
This diploma thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the diploma degree in Software Engineering. The thesis is equivalent to 10 weeks of full time studies.

Contact Information:
Author(s):
Tobias Almroth
E-mail: tobalm.vim@gmail.com

University advisor:
Kennet Henningsson
Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Internet: www.bth.se
Phone: +46 455 38 50 00
Fax: +46 455 38 50 57
Abstract

This paper looks into how data is best visualized and how visualizations should be designed to be most easily perceived. Furthermore the study looks into what tools there are available on the market today for visualizing data in angular 5 applications. With regards to a client, a developer team from the swedish police IT-department, the tools are evaluated and the one most suitable for the client is found. The paper also looks into how a dynamic data solution can be developed in angular 5. A solution where data can be selected in one component and displayed in another.

To answer the questions sought a study of previous research into data visualization was done as well as a look into how angular 5 applications can be developed. Interviews with the clients were held where their specific requirements on visualization tools were identified. After searching and listing available visualization tools on the market the tools were evaluated against the clients requirements and a prototype application were developed. Showcasing both the most suitable tool and its integration but also a dynamic data solution in angular 5.

As a conclusion data visualizations should be made as simple as possible with the main focus on the data. When it comes to tools the one most suitable to the client was Chart.js that easily integrated into an angular 5 application. An application that thanks to angular’s features is well equipped for handling and developing dynamic data solutions.

Keywords: Data visualization, JavaScript, Angular 5, Web application
Content

Abstract 1

Content 2

1. Introduction 4
  1.1 Background 4
  1.2 Purpose 4
  1.3 Scope 5

2. Questions to be answered 5
  2.1 What makes a good visualization of data? 5
  2.2 What visualization tool provides the best fit for the client? 5
  2.3 What is a good solution for dynamic data selection? 5

3. Method 6
  3.1 Literature study 6
  3.2 Interviews & evaluation 7
  3.3 Implementation 7

4. Results and analysis 8
  4.1 Data visualization 8
    4.1.1 Graphs 8
    4.1.2 Design considerations 11
  4.2 Visualization tools 12
    4.2.1 Interview results 12
    4.2.2 Search 13
    4.2.3 Evaluation 14
    4.2.4 Final decision 17
  4.3 A dynamic data visualization solution in Angular 18

5. Conclusion 24

6. Future work 25

7. References 26

8. Appendices 28
  1. Data visualization tools 28
  2. Initial elimination 31
  3. Not free for commercial use 32
  4. Frameworks not available on NPM 33
5. Last commit and contributors 33
6. Interview questions and answers 34
1. Introduction

This is a study into data visualization, which ones work best for what kind of data, what tools there are available and which is most suitable for a client team. It is also a look into how a data visualization solution can be implemented into said teams working environment with regards to their technology and developing tools.

1.1 Background

The client is a team of developers IT-department of the swedish police. The team currently develops, maintains and manages several computer systems.

Most of the software the team handles are constructed using Java technology. Both on the back-end and the front-end. The applications range from native Java applications to web based Java applications. Now the team is in process of switching technologies towards a more future-proof web based technology stack. The front-end side of which will be based on Angular 5, a JavaScript based framework.

In conjunction with the move to a more web and angular based stack they are also interested in seeing what possibilities there are to visualize data. In the past when data/statistics have been requested the team has delivered this in a datafile straight from a database. Now they are interested in seeing what possible solutions for visualizing data there are in web based applications and which would be the most suitable for their uses.

1.2 Purpose

The purpose of this study is to explore what knowledge surrounding data visualization exists. What are the considerations to make when creating such a solution. When do we use which type of graph and what design considerations should be taken into account.

The second purpose of the study is to come up with a good solution for visualizing data in angular 5 based web applications. What graph framework is the best for this teams requirements and would be the most suitable for use in future projects.

Furthermore how can we create a dynamic data visualization solution where we give the user a chance to select the data he/she require. Since the data sets easily can grow in size the need to filter and/or select specific data to visualize is requested.
1.3 Scope

The study will focus on looking into the foundational forms of visualizations. Consisting of the main types of diagrams widely known. As designers and data visualization have evolved over the more recent years many new types of visualizations have arisen. Heat-maps and other more graphical variants for more advanced types of visualization are now available. This however lay outside the scope of this study that focuses on the foundations.

The scope of the study regarding visualization tools is set to review JavaScript based charting libraries and visualization solutions that can be used in angular 5 applications. The focus will be on tools that are freely available on the market and where the softwares licenses make them free to use in a commercial setting.

2. Questions to be answered

2.1 What makes a good visualization of data?

The goal of this question is to look into how data should be presented to be most easily perceived. In the end a conclusion will be reached as to what type of graph should be used to present which type of data and if there are any design considerations to take into account.

The value of this question will be beneficial in future work as to showcase what should be taken into account when visualizing data. The answer to this question will benefit both the student, client and other readers who are about to visualize data.

2.2 What visualization tool provides the best fit for the client?

The goal is to evaluate available tools for data visualizations and to come up with a conclusion as to which tool would be the most suitable for the client to use in future projects.

The main beneficiary of the result of this question is of course the client, but also the student and others who are interested in getting an overview of what tools are available on the market.

2.3 What is a good solution for dynamic data selection?

Instead of having separated pages or fixed images for each visualization it would be better for the user to be able to select which visualization and what data it wants to view.

The answer to this question will lead up to a modular, extendable solution to allow data to be dynamically selected and used in the visualization. The goal is to make it easy to extend the application with new visualization data without making large revisions of the code.
The beneficiary here is both the client who easily will be able to extend the application with new visualization data and the user who in this case easily and quickly will be able to receive new visualizations as per their need.

3. Method

To come to a result and conclusion into what constitutes a good data visualization, what tool is the best fit for the client team and how a good dynamic solution in angular can be constructed, a few different methods were used. First and foremost a literature study was performed to get an understanding of the current findings in the field and to gather knowledge about the study's focus areas. After that interviews were held with the client team to determine their requirements and what demands they have on a prospective tool. The results of the interview were then used in an evaluation of possible tools. Finally an implementation according to the clients requirements were conducted and a prototype was developed showcasing a possible solution for the visualization of data in an environment similar to the clients.

3.1 Literature study

For this part of the study and to find the answer to the first of the study's questions a study of available literature was made to gain knowledge regarding the current best practices when it comes to visualizing data. A search was made at the library, through BTH’s library’s “summon” database and also using Google for relevant data regarding how data is best visualized and what graph is best used for what type of data.

At the library a search through the computer science section and the section for UX/UI design I found the book “Information Dashboard Design: Displaying data for at-a-glance monitoring”. A book that goes into great detail about designing graphs and different graphs use cases. The author, Stephen Few, has been working with data visualization for more than 30 year and I have seen references made to his works on several places on the internet. [5]

I also visited the website of SCB (the swedish department of swedish statistics). There I found some basic information regarding different graphs.[1]

The next step was a look at SWEBOK, a document published by IEEE (a non-profit organisation that collects and publishes facts about technology) and it’s chapter on software design that went over how color can be used to enhance the user experience.[2]

I also did a search on Google of “data visualization” and found an article by the NN Group, a group that does research on UI/UX design, that discussed the use of different kinds of graphs and visualizations and their use with colors and 3d.[4] My google search also revealed an article published by the Harvard Business Review written by Scott Berinato, an author of books on data visualization, that detailed that data visualization is more than just making some graphs.
3.2 Interviews & evaluation

To answer the study's second question interviews were conducted with the members of the client developer team. Through the interviews knowledge of the teams requirements and demands regarding visualization tools was learned.

Prior to conducting the interviews a search into previous evaluation of similar tools was made. From that an understanding into what possible requirements and features were possible was gained. That understanding formed a baseline from which the interview questions were constructed.

The interviews were conducted using different types of questions and after all questions were answered a validation of the findings were made with each subject, wherein they were allowed to read through the answers to validate that the result of the interview corresponded with the subjects views.

After the interview process a search of available tools was made through Google. A list of available possible tools were established for further evaluation.

Each tool was then evaluated based on the client teams requirements and demands. This process reduced the list of possible tools down to a handful that then went through to be evaluated further. The remaining tools was then taken into a process of trying to implement them into a prototype application similar to the applications the team are developing. During this process a few of the tools were discarded due to lacking good enough documentation and/or community rendering the process of integrating them to problematic.

After the prototype application was completed and all tools had been assessed a final conclusion of the most optimal tool for the clients purposes was reached.

3.3 Implementation

To develop a modular solution that would allow for dynamic selection of data using the client teams technology an exploration of the angular 5 framework and some of it’s incorporated dependencies was done.

The research of this was primarily made on the angular 5 website and on the website of RxJS, a dependency used by angular and that helps handling data streams.[6, 7]

This exploration lead to insights into the framework and how data might be passed from components to each other. As well as some principles of how data delivery from database to application might be constructed. Using these insights a prototype was constructed that allow the user to dynamically select data and visualization. The data is then processed to fit the visualization and passed to the visualization component allowing the visualization to be rendered.
4. Results and analysis

4.1 Data visualization

Visualizing data well is becoming increasingly important. As more and more decisions are based on a foundation of data the ability to visualize this data well to aid in the decision making process is becoming a highly sought after skill. This in combination with the increasingly large amount of data that's being collected has made data visualization more complex than just putting together some graphs.[3]

A good visualization of data is simple, clear, accurate and lets the recipient perceive the message even before paying attention to it. It only takes a few seconds to digest a good visualization where as a bad one will cause new questions and take longer to interpret, ultimately wasting time instead of saving it. [3, 5]

When visualizing data there are a number of considerations to make. Firstly one should consider the purpose of the visualization, what message should be conveyed. Here one should strive for simplicity, preferably a visualization should only display a single message using as few variables as necessary.[3]

The second consideration to make is to choose the right graph for the job. When choosing the type of graph for your visualization to goal should be to choose the best graph for the job. It might be attractive to select a graph type that is popular for the time being or looks in a certain way. The rule however is to look past this. Trends and designs change but the purpose of the visualization, to convey the message, will be constant.[5]

A final consideration is in the visual design of the visualization. This is a two part section. The visualization should be pleasing to look at, an aesthetically pleasing experience will excite users. The inverse is also true. A bad visual appearance will undermine the experience and make users leave.[5]

4.1.1 Graphs

To visualize data there are a number of different styles and designs to choose from. Theses types, even though they may look different, usually have their foundation in a few different types of graphs. There are a number of different types of graphs and choosing between them is not always easy. Which graph is more suitable to what type of data and are there any special considerations to take into account.

The goal when choosing a graph should be to choose the best type for the data that is to be visualized.[5]
Bar graph
The bar graph is one of the most useful graph types. It is used for displaying multiple values from a single category like a company's sales per weekday. The bar graph is a good choice when you want to display quantitative data and also need to know precise values and how they relate to one another in the category. The length of the bars makes it easy to compare differences in values. Furthermore the bars of the graph can also be subdivided to show multiple sub values, either in a stacked bar chart or in multiple columns stacked close to each other.[1, 4, 5]

The bar graph is also, contrary to the popular pie graph, the most efficient way of displaying different data as part of a whole. For instance seeing how much of total weekly sales come from which weekday. The human mind has an easier time of distinguishing between the height of bars than estimating the area of slices in a pie graph. However, when used in this way, due to the prevalent choice of pie graphs for this application it is important to make it clear to the recipient that the data is visualized in this manner.[5]

When designing your bar graph there are a few considerations that should be made. The bars should always be of the same width and the scale on the quantitative axis should always start at zero. This is to make it easier to interpret the amount of data the bars represent. To make the graph even more clearer and to help make it easier to see the difference in values the bars can be aligned in an ascending or descending order.[1, 4, 5]

Line graph
The line graph is used to show variations of data over time. It makes it easy to see how a value changes over time and the rate of the change.[1, 5]

Considerations for the line graph are to keep the distance between points the same between all data points so as to not make the variation seem different. As opposed to the bar graph here the quantitative axis is not required to start at zero since were less interested in the value in itself than the actual change. Furthermore the number of lines shown in a single graph should not exceed 5 as that will cause to much clutter. The lines color should also be sufficiently different to make them stand out individually.[1, 5]
Pie graph
The pie graph shows data as parts of a whole. Each slice is equivalent to percentage equivalence of the whole.[1]

Even though the pie graph is one of the most common graphs used it’s, contrary to belief, one of the harder to understand and interpret. Since the pie graph conveys the size of the data using area it makes it more difficult to interpret and compare the values to each other than a bar graph. This makes the pie graph only really useful when comparing values that have large differences.[4, 5]

Radar graph
The radar graph is a mix of a pie graph and a line graph that’s surfacing more and more. In its core it is a line graph with a circular category axis. The circular nature of the graph makes it more difficult to interpret and compare values than a basic line graph.[5]

Scatter plot & bubble plot
The scatter plot uses dots to display the correlation between 2 different values. It makes it easy to see different trends and what of the the 2 values that is causing it.[5]

The bubble plot is basically a scatter plot where a third variable has been added. This variable is visualized using the area of the dot.[5]
4.1.2 Design considerations

Data visualization comes in a lot of different forms and designs. For them to be as effective as possible at conveying their message it is worth thinking about how they should be designed. Make sure your visualization is not unattractive, but for the sake of that don't overuse visual effects. Things like 3D, background images or too many vibrant colors directs attention away from what's important. Think in the lines of simplicity, clarity and accuracy.[5]

The focus should be on the data. Thus the data-pixels, like bars, lines etc., should be the most prominent. The non-data pixels like gridlines, borders, titles and headings should be muted so as not to draw attention away from what's important. A good way of doing this is to lower their brightness making them lighter.[2, 5]

The same goes for colors. Use muted colors that don't draw too much attention to the color in itself. Bright vibrant colors are more exhausting and demands too much attention. Also when it comes to the use of colors, use them not to signal measures of quantity but rather to signal what belongs together or as a means to draw attention to something special. For the sake of users with different disabilities, like color blindness, make sure that when different colors are used that their hue and or contrast also are different.[4, 5]
4.2 Visualization tools

For this part of the study a series of interviews were conducted with all of the current developers of the client developer team which at the time amounted to 3 in total. All 3 developers interviewed have many years experience in developing and managing enterprise level applications in commercial and government environments. The developers mainly work in Java as their main language and have no to very limited prior knowledge of Angular 5.

The interview questions and the subjects results can be viewed in appendix 6.

4.2.1 Interview results

After conducting the interviews a summary of the results were made and from that summary the requirements for evaluation were formed. The subjects were all pretty much conclusive in their answers to most of the questions. In that case the requirements were easily formed. In a few instances the subjects views differed and in those cases the requirement were formed using the answers that numerously mostly were the same. These requirements formed the basis of the evaluation.

One opinion that all the subjects shared were that inhouse knowledge amongst other teams in the police department about tools would be a valuable asset. However due to time-constraints a survey and/or interviews with other teams were not possible and thus this possible requirement were omitted.

License

The team prefer to initially use software with a license that makes it free to use in a commercial setting. This is mainly due to the long administrative process of purchasing software.

Software source

To be able to use and download the software as part of a build process the team requests that the software is available to download through NPM, a package manager for JavaScript packages. Preferably there was also no external complicated dependencies requiring extra installation.

Compatibility

The target environment the tool will be used and integrated into will be an angular 5 application. The application and the tool itself must support the browser Internet Explorer 11 and later versions. The tools must be able to function in an environment without connection to the internet.

Responsiveness

Since the team develop their software for a closed group of users they also know their users working environment. For the foreseeable future the applications intended for data visualization will be desktop based. They will be created to use “full screen” and thus responsiveness is not a requirement. That being said responsive support could become useful.
Stability, size & maintenance
A key requirement of the tool to be chosen was that it must be stable, not an alpha or beta version, and continually be maintained and updated. Furthermore it was also preferred that the tool had a large user base as well as several people that contribute to the tools code. Having a large user base would lead to a large active community and thus it would be easier to find information regarding solutions and problems. The same can be said about the number of contributors to the code. The team specified that evaluating this was usually done looking at GitHub and NPM statistics like stars, followers, resolved issues and then number of downloads.

Documentation
The tools documentation was also of a high priority. A good documentation makes it easier to quickly start using it. The documentation should also preferably have plenty of examples and/or guides to facilitate quick start p of development.

Graph types
The team concluded that a basic range of graph types would be sufficient. Graphs in this category is line, bar, pie, radar, scatter and bubble graphs.

Interactivity
The ability to get tooltips and information of data points on mouse over was something the team requested. The same with the possibility to bind events and functions to mouse clicks. The ability to zoom could be seen as a nice to have feature but not something that was required.

Design
When it came to design the team thought that a simple, clear and appealing design was nice to have and that it should be possible to customize the looks. They did however conclude that functionality was a higher priority than a flashy design.

4.2.2 Search
The search for possible tools to evaluated was performed using google. The search-terms used were “javascript charts” and “javascript graphs”. The search yielded in a lot of different tools and also several articles containing comparisons and collections of tools, as well as a wikipedia entry listing many tools.[12, 13, 14, 15, 16, 17] All tools found in the references listed were added to the list and scheduled for evaluation.

The result of the search rendered in a list of 67 different tools that can be used for visualizing data and are based on JavaScript.[appendix 1]
4.2.3 Evaluation

The evaluation process began by visiting each tools website and verifying that the tool would be suitable for use in the target environment. This part corresponded with the compatibility requirements established during the interviews. Quite a few tools were immediately eliminated due to a number of reasons. Some of the tools had websites that were not in English or Swedish and thus could not be understood. A few of the tools were online tools only and a few tools were part of larger toolkits/frameworks. Quite a few of the tools eliminated were also done so because they were developed for use with specific technology, such as PHP and React, that was not inline with the target environment.

In total 23 tools were eliminated during this round. [appendix 2]

The next step looked at the tools licenses. From the interviews it had been learned that the tool must be free to use in a commercial setting and thus tools that failed this requirement were discarded. Some tools that were free to use with watermarks and linkable logos were also eliminated. This round eliminated 18 tools. [appendix 3]

The next step was to verify that the tools were available at GitHub and NPM. This was a prerequisite to be able to evaluate the tools stability, maintenance and community, all important requirements laid out by the interviewed subjects. 8 tools failed to meet this criteria and was thus eliminated. [8, 9, appendix 4]

After the initial evaluation 19 tools remained. At this point statistics were gathered on the remaining tools using GitHub, NPM and a site that displays download statistics from NPM. At GitHub the initial screening was to look at when the last commit to the software was made. 4 tools were eliminated since they hadn’t had a commit done to them over the past year. After this the tools with less than 10 contributors were also eliminated. This resulted in 8 tools being eliminated. [8, 9, appendix 5]

So far in the evaluation process 11 tools remained. Looking at other statistics revealed that they all had good numbers. Using the site npm-stat.com statistics regarding how many downloads each tool had over the past prior year was collected. The result ranged from just over 7000000 to about 35500. Though being a large difference they all seem to see plenty of use. [9]

Looking at GitHub data regarding stars (equal to a repository being liked and followed), forks (the repository being downloaded for modification) and open issues and pull requests revealed that all but one of the remaining tools had sufficient statistics indicating that they all have a large user base and following. One of the tools were standing out in the number of open issues, echarts, with more than 3 times the number of the second in order. Upon further inspection of the repository it was revealed that the software seem to be in a state of heavy development and most of the information regarding issues were in Chinese. Since it was hard to evaluate what the nature of theses issues indicated the tool was eliminated from further evaluation. This elimination corresponded with the requirement that the tool should have good documentation and be sufficiently stable to suit the client team.
Remaining at this point was 10 tools.

- Chart.js
- C3.js
- NVD3
- Chartist
- Plotly.js
- VisJS
- Dc
- Plottable
- Taucharts
- MetricsGraphics

At this point a comparison of the feature requirements of the client team and the tools was made regarding graph types and interactivity in the graphs. The requirements sought were the ability to construct basic graphs (line, bar, pie, scatter) and the features of tooltips (mouseover popups) and the ability to capture click events from the graphs.

MetricsGraphics was eliminated due to the lack of click event functionality. Chartist and taucharts were eliminated due to the need for third party plugins to enable tooltip/mouseover functionality and VisJS was eliminated due to lacking the functionality to draw scatter plots. Remaining was 6 tools that would go on to be integrated into a prototype angular 5 application similar to what the client would be developing.

**Prototype integration**

The final step for evaluation was to satisfy the client requirements on good documentation and ease of integration into the development environment that constituted an angular 5 application. For this part a prototype angular application was constructed. Then the remaining tools were attempted to be integrated into the application. To aid in this process the respective tools documentation, including guides and examples, that was available on respective tools website was consulted. When a problem arouse in this process a google search for a solution was made using the keywords, [tools name], angular and [problem]. This process detected very well each tools suitability from an ease of development standpoint as well as clarifying the tools community status from an angular development perspective.

**C3.js**

The integration went relatively easy thanks to a basic guide in the documentation. Decent guide to get started in the documentation and plenty of examples regarding different graph types. A small issue was quickly resolved with a search on google that almost immediately returned a solution on Stack-overflow. This tool also has a simple basic design with possibilities to configure extensively.
Chart.js
This tool had very good documentation with plenty of guides and examples that made integration straight forward. No issues were encountered during the process. The tool has large possibilities for configuration and adaptation both feature wise and design wise with a good looking overall design functionality.

Dc.js
The integration process of dc.js into the application failed due to a few different reasons. First of all the tool lacks good documentation and guides to get started. Searching for help using google did not help in the matter and during the process it was revealed that the tool depends on several other libraries that now are deprecated and out of date. Thus the library was eliminated from further review.

NVD3
During the attempt to integrate NVD3 several issues were realized. The biggest of which was the tools dependency on a deprecated version of a library called D3. Other issues that could not be resolved, even through extensive reading of the documentation and through search through the community was type errors during compilation. Furthermore the library requires extensive knowledge of its dependency D3 to utilize. Lacking documentation, deprecated dependencies and a small community for angular contributed to eliminating this tool from evaluation.

Plotly.js
Plotly.js could failed to completely integrate with our prototype application. Several type errors occurred at compilation time and during problem solving and investigating their cause it appears that the tool is not fully compatible with angular without extensive modification to the angular build process itself or without restricting the tools functionality. Though the documentation is extensive and has a good amount of examples it is primarily focused on use in standalone vanilla javascript environments and not on angular based applications or similar. This was also true regarding search of possible solutions throughout the community.

Plottable
Plottable could failed to integrate into our application due to rendering errors. Furthermore the documentation is not updated to mirror the latest version of the tool and could thus not be used to help solve the problems. Searching for a solution throughout the community also failed to reveal a solution. In fact the community around Plottable and towards angular in particular form a discussion standpoint was found to be very limited.
4.2.4 Final decision

Throughout this evaluation of data visualization tools with a focus on use in angular 5 based applications I started with 67 different tools. After evaluation the tools and comparing their features and characteristics against the specified requirements of the client 2 tools were found to live up to to the requirements. C3.js and Chart.js. Both tools integrated easily and quickly into the prototype application.

Both tools fulfill all the requirements specified. One of the tools eases ahead of the other slightly. Chart.js has better documentation and what appears to be a larger community. It also has a more aesthetic look out of the box and provides a more modern and visually appealing design. Thus the winner and/or tool recommended for the team to use in future projects is Chart.js.
4.3 A dynamic data visualization solution in Angular

Here we implement a prototype solution for displaying and selecting data dynamically in an angular application. The data is fetched from a back-end REST API and then parsed and displayed in the application.

Angular

Angular is a JavaScript framework used to build single page applications developed by Google. The framework is comes organized into several libraries providing all base functionality needed to build and develop single page applications. Functionality like making http requests, handling routing and form validation etc.

Angular is built using TypeScript, a superset of JavaScript, that facilitates the use of defined types for objects and variables etc.

The framework is built up of modules that collects various resources and functionality. An application has at least one module, the root module called AppModule, containing the application. Within this module other modules may be imported providing separate functionality and/or features.

Within the modules there are components. Components are used to define views or smaller reusable parts of views. Each component has its own data, logic code and template and style code that is scoped to that particular component. A component can be anything from as small as a button up to a whole page/view.

For handling other functionality that is not specific for displaying the view components use services. Services can do a multitude of things. Storing data, making http requests, be used for handling component to component communication among other things.

The prototype

The prototype will be used to allow the user to select different types of data to be visualized. The design of the prototype may look quite basic but each component can easily be imported and reused in other components.
For the sake of rapid prototyping and applying basic usability and design bootstrap 4 default theme will be used.

The prototype will have 3 components making up our views of the application.

- AppComponent - our root component that contains our application
- DataSelectComponent - a form allowing the user the ability to select the requested data
- DataDisplayComponent - containing the visualization of the selected data.

Furthermore the prototype will also have a service that handle requests to our back-end API and passing data between our components.

- DataService

AppComponent

This will be the root component of the application and will be the first component loaded.

Here we include and place our other components, the DataSelectComponent and DataDisplayComponent.

No special logic will be placed in AppComponent.

DataSelectComponent

In this component we will have a select element where the user can select the data he/she wishes to display.

Here in the template we also bind a call to the method “onSelectData()” on the change event generated from the select element.

In our data-select.component.ts file we first begin with importing all dependencies required by the component.
In the constructor we inject our data service so it can be used in the component.

The method “ngOnInit()” is an angular life-cycle method and will run as soon as the component have been initialized. Here we call and subscribe to our data service and its method “getDataFromApi()” to make a call to our back-end and get our data sets.

Subscribing to methods that return observables is widely used in angular development and part of the reactive library RxJS that comes bundled with angular. It establishes a connection to the observable and waits for data that can come one or more times. When data is then received a callback function is executed.

The data from the back-end is then stored in the private variable “apiData” inside the component.

As mentioned earlier when the user selects a data set in the dropdown select in the template the method “onSelectData()” will be run. Here we parse out which data set that was selected and then send the specific data set as an argument to our DataService “sendData()” method. This method will then send the data set onwards to our data display component.
Data display component

The template for the data display component is quite minimal. It’s basically a placeholder for our graph/chart. It uses one of angular’s built in directives “*ngIf” to check if a chart exists and if so display it.

In our .ts file and the “ngOnInit()” method we create a subscription to the data service and its “getData()” method listening for any data being sent from our data select component.

When data is received it is then passed on to the “createChart()” method that creates and displays the data using the visualization tool Chart.js.
Displaying the data

When our data display component receives data to be displayed it passes it to the “createChart()” method. The method uses Chart.js to create a new chart object with a reference to our canvas element in the template with id “lineCanvas”. Passed in to the chart object is a JSON-object containing various configuration for the chart as well as the data to be visualized. This chart is then stored in the component variable “lineChart”.

This will display a visualization/chart with the requested data.

If a new data set is selected this method will be called again with the new data and the chart will be reinitialized.
**Component to component communication**

In angular there are several ways of sending data between components. One of the easier and more modular solutions is via the use of a service and something called subjects and observables.

These are part of the reactive library RxJS. Observables are an object type that can be subscribed to. The subscription creates a connection to the observable through which data can be passed and acted upon.

Thus we can create subscriptions between components and services where the components wait on data to be passed.

Subjects and behaviour subjects are a type of observable that can be subscribed on. The difference between a subject and a behaviour subject is that the later has an initial value where as a subject does not. Meaning that when a subscription is made to a subject no value will be received from the start until the subjects “next()” method is called.

In our service we have 3 methods.

- **getDataFromApi()** - uses angular built in http client for making http requests. It returns an observable which our components use to subscribe to and receive data from our API.
- **getData()** - returns a subject as an observable letting components subscribe to it and letting us send data to all components that are subscribing to it.
- **sendData()** - uses the subjects “next()” method to send data to the subjects subscribers.

This is all that is needed to get data from our API and send and receive data from one component to another.
5. Conclusion

This study set out to answer 3 different questions regarding data visualization.

The first part of the study looked into how data should be visualized to be most easily perceived, trying to answer the question: What makes a good visualization of data?

After reviewing current findings it can be concluded that a few basic types of graphs go a long way when it comes to visualizing data. The bar graph is the goto choice when a single category is what is to be visualized. It is also preferable over the popular pie graph when it comes to displaying data as part of a whole. The line graph is the choice when it comes to visualizing data over time and if there is a need to display the correlation of more data points either the scatter plot (2 values) or the bubble plot (3 values) are good to use. The design of the graphs themself also matter. A to distracting design of the graphs can detract attention from what should be the main focus, the data. The design should thus be focusing on the data, emphasizing it in a subtle way, and muting the non data like grid lines etc.

This study also analysed the requirements of a developer team and evaluated a plethora of different visualization tools against these requirements. Here the study aimed to answer the question: What visualization tool provides the best fit for the client?

Since there is a substantial amount of tools available on the market a proper analysis of the requirements for a tool is important to make before trying to select the right one for the job. Many of the tools differ in small ways and some are better than others when it comes to displaying certain types of data or working in a specific environment. The tool that provided the best fit for the client in this study was Chart.js. The tool fulfills all requirements as laid out by the client. It is a stable open-source tool easily available from NPM or GitHub with a large community and that is being frequently updated and maintained. It has all the features the client requested like available graph types, graph interactivity and a customizable attractive design.

The final part of the study looked into how a dynamic modular application can be constructed in angular 5. The goal here was to showcase and answer the question: What is a good solution for dynamic data selection?

The study here shows a good example of an implementation of Chart.js into an Angular 5 application. An application with different components that allow a user to select certain data to be visualized in one component and where the data then will be shown in another. For this purpose as shown angular provides several built in usable features and is a good and fitting tool for the task at hand.
6. Future work

This study has looked into the basic foundations of data visualization that exists, straightforward graphs of different kinds. As a continuation of this and with the growing focus on data and statistics one could expand this and do a comparison of the more fundamental graphs discussed in this study and with more elaborate and visual types of visualizations often containing images and figures of different kinds. Further study of the ease of perception of these newer and more elaborate visualizations versus the foundational graphs would also be interesting. How easy are the newer ones to understand and interpret and are they good solutions or should one stick to the true and trusted graphs that have been around for a long time.

When it comes to evaluating different tools it is a constantly growing market with constantly changing tools that evolves and thus fits different technology and requirements. It is important to not become comfortable with a certain tool used in a specific project. As technology and projects change it is important to reevaluate the available options to find a fitting solution.

When it comes to building modular components that interact with each other we can see that angular comes with a lot of built in functionality to facilitate this and that makes it easy to develop a good solution. Further study in this field would be to compare how other frameworks stack up to angular. How can they be used for similar purposes and which framework if any would be more optimal for these types of applications.
7. References


8. Appendices

1. Data visualization tools

<table>
<thead>
<tr>
<th>Framework</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>amCharts</td>
<td><a href="https://www.amcharts.com/download/">https://www.amcharts.com/download/</a></td>
</tr>
<tr>
<td>AnyChart</td>
<td><a href="https://www.anychart.com/buy/">https://www.anychart.com/buy/</a></td>
</tr>
<tr>
<td>C3.js</td>
<td><a href="http://c3js.org/">http://c3js.org/</a></td>
</tr>
<tr>
<td>CanvasJS</td>
<td><a href="https://canvasjs.com/license/">https://canvasjs.com/license/</a></td>
</tr>
<tr>
<td>Chart Builder</td>
<td><a href="http://www.livegap.com/charts/">http://www.livegap.com/charts/</a></td>
</tr>
<tr>
<td>Chart.js</td>
<td><a href="http://www.chartjs.org/docs/latest/">http://www.chartjs.org/docs/latest/</a></td>
</tr>
<tr>
<td>Chartist</td>
<td><a href="https://gionkunz.github.io/chartist-js/">https://gionkunz.github.io/chartist-js/</a></td>
</tr>
<tr>
<td>chartkick.js</td>
<td><a href="https://github.com/ankane/chartkick.js">https://github.com/ankane/chartkick.js</a></td>
</tr>
<tr>
<td>Charts 4 PHP</td>
<td><a href="http://www.chartphp.com/">http://www.chartphp.com/</a></td>
</tr>
<tr>
<td>cubism.js</td>
<td><a href="https://square.github.io/cubism/">https://square.github.io/cubism/</a></td>
</tr>
<tr>
<td>cytoscape</td>
<td><a href="http://js.cytoscape.org/#introduction">http://js.cytoscape.org/#introduction</a></td>
</tr>
<tr>
<td>d3.js</td>
<td><a href="https://d3js.org/">https://d3js.org/</a></td>
</tr>
<tr>
<td>dc</td>
<td><a href="https://dc-js.github.io/dc.js/">https://dc-js.github.io/dc.js/</a></td>
</tr>
<tr>
<td>DevExtreme</td>
<td><a href="https://js.devexpress.com/Buy/">https://js.devexpress.com/Buy/</a></td>
</tr>
<tr>
<td>DHTMLMX Charts</td>
<td><a href="https://dhtmlx.com/docs/products/licenses.shtml">https://dhtmlx.com/docs/products/licenses.shtml</a></td>
</tr>
<tr>
<td>dimple</td>
<td><a href="http://dimplejs.org/">http://dimplejs.org/</a></td>
</tr>
<tr>
<td>Dojo Charting</td>
<td><a href="https://dojotoolkit.org/documentation/tutorials/1.10/charting/index.html">https://dojotoolkit.org/documentation/tutorials/1.10/charting/index.html</a></td>
</tr>
<tr>
<td>Dygraphs</td>
<td><a href="http://dygraphs.com/">http://dygraphs.com/</a></td>
</tr>
<tr>
<td>EJS Chart</td>
<td><a href="https://github.com/EmpriseCorporation/EJSCharts">https://github.com/EmpriseCorporation/EJSCharts</a></td>
</tr>
<tr>
<td>elycharts</td>
<td><a href="https://elycharts.com/features">https://elycharts.com/features</a></td>
</tr>
<tr>
<td>ember charts</td>
<td><a href="http://opensource.addepar.com/ember-charts/#/overview">http://opensource.addepar.com/ember-charts/#/overview</a></td>
</tr>
<tr>
<td>Factmint Charts</td>
<td><a href="http://factmint.com/charts">http://factmint.com/charts</a></td>
</tr>
<tr>
<td>Flot Charts</td>
<td><a href="https://www.flotcharts.org/">https://www.flotcharts.org/</a></td>
</tr>
<tr>
<td>Flotr2</td>
<td><a href="http://www.humblesoftware.com/flotr2/">http://www.humblesoftware.com/flotr2/</a></td>
</tr>
<tr>
<td>FusionCharts</td>
<td><a href="https://www.fusioncharts.com/buy/annual/">https://www.fusioncharts.com/buy/annual/</a></td>
</tr>
<tr>
<td>Google Charts</td>
<td><a href="https://developers.google.com/chart/">https://developers.google.com/chart/</a></td>
</tr>
<tr>
<td>Library</td>
<td>URL</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>gRaphael</td>
<td><a href="http://g.raphaeljs.com/">http://g.raphaeljs.com/</a></td>
</tr>
<tr>
<td>Highcharts</td>
<td><a href="https://www.highcharts.com/products/highcharts/">https://www.highcharts.com/products/highcharts/</a></td>
</tr>
<tr>
<td>JenScript</td>
<td><a href="https://jenscript.io/">https://jenscript.io/</a></td>
</tr>
<tr>
<td>jqPlot</td>
<td><a href="http://www.jqplot.com/">http://www.jqplot.com/</a></td>
</tr>
<tr>
<td>jQuery sparklines</td>
<td><a href="https://omnipotent.net/jquery.sparkline/#s-about">https://omnipotent.net/jquery.sparkline/#s-about</a></td>
</tr>
<tr>
<td>js charts</td>
<td><a href="http://www.jscharts.com/free-download">http://www.jscharts.com/free-download</a></td>
</tr>
<tr>
<td>KoolChart</td>
<td><a href="https://www.koolchart.com/buy/">https://www.koolchart.com/buy/</a></td>
</tr>
<tr>
<td>MetricsGraphics</td>
<td><a href="https://www.metricsgraphicsjs.org/">https://www.metricsgraphicsjs.org/</a></td>
</tr>
<tr>
<td>morris.js</td>
<td><a href="http://morrisjs.github.io/morris.js/">http://morrisjs.github.io/morris.js/</a></td>
</tr>
<tr>
<td>n3 charts</td>
<td><a href="https://n3-charts.github.io/line-chart/#/home">https://n3-charts.github.io/line-chart/#/home</a></td>
</tr>
<tr>
<td>NextCharts</td>
<td><a href="https://github.com/nextreports/nextcharts">https://github.com/nextreports/nextcharts</a></td>
</tr>
<tr>
<td>NVD3</td>
<td><a href="http://nvd3.org/">http://nvd3.org/</a></td>
</tr>
<tr>
<td>OLAPCharts</td>
<td><a href="https://www.kyubit.com/Kyubit-Business-Intelligence-Purchase">https://www.kyubit.com/Kyubit-Business-Intelligence-Purchase</a></td>
</tr>
<tr>
<td>plotly.js</td>
<td><a href="https://plot.ly/javascript/">https://plot.ly/javascript/</a></td>
</tr>
<tr>
<td>plottable.js</td>
<td><a href="http://plottablejs.org/">http://plottablejs.org/</a></td>
</tr>
<tr>
<td>PlusCharts</td>
<td><a href="http://www.pluscharts.com/">http://www.pluscharts.com/</a></td>
</tr>
<tr>
<td>protovis</td>
<td><a href="https://mbostock.github.io/protovis/">https://mbostock.github.io/protovis/</a></td>
</tr>
<tr>
<td>ReactiveChart</td>
<td><a href="https://reactivechart.com/">https://reactivechart.com/</a></td>
</tr>
<tr>
<td>RGraph</td>
<td><a href="https://www.rgraph.net/download">https://www.rgraph.net/download</a></td>
</tr>
<tr>
<td>rickshaw</td>
<td><a href="http://code.shutterstock.com/rickshaw/">http://code.shutterstock.com/rickshaw/</a></td>
</tr>
<tr>
<td>Shield UI</td>
<td><a href="https://www.shieldui.com/purchase-packages">https://www.shieldui.com/purchase-packages</a></td>
</tr>
<tr>
<td>sigmajs</td>
<td><a href="http://sigmajs.org/">http://sigmajs.org/</a></td>
</tr>
<tr>
<td>smoothie charts</td>
<td><a href="http://smoothiecharts.org/">http://smoothiecharts.org/</a></td>
</tr>
<tr>
<td>Syncfusion</td>
<td><a href="https://www.syncfusion.com/products/essential-studio">https://www.syncfusion.com/products/essential-studio</a></td>
</tr>
<tr>
<td>taucharts</td>
<td><a href="https://www.taucharts.com/">https://www.taucharts.com/</a></td>
</tr>
<tr>
<td>TeeChart JS</td>
<td><a href="https://www.steema.com/product/html5#pricing">https://www.steema.com/product/html5#pricing</a></td>
</tr>
<tr>
<td>toast ui chart</td>
<td><a href="https://github.com/nhnent/tui.chart">https://github.com/nhnent/tui.chart</a></td>
</tr>
<tr>
<td>uvcharts</td>
<td><a href="https://www.npmjs.com/package/uvCharts">https://www.npmjs.com/package/uvCharts</a></td>
</tr>
<tr>
<td>Vaadin Charts</td>
<td><a href="https://vaadin.com/pricing">https://vaadin.com/pricing</a></td>
</tr>
<tr>
<td>VanCharts</td>
<td><a href="http://www.vancharts.com/buy.html">http://www.vancharts.com/buy.html</a></td>
</tr>
<tr>
<td>victory</td>
<td><a href="https://formidable.com/open-source/victory/">https://formidable.com/open-source/victory/</a></td>
</tr>
<tr>
<td>VisJS</td>
<td><a href="http://visjs.org/">http://visjs.org/</a></td>
</tr>
<tr>
<td>Webcharts</td>
<td><a href="https://openresearchsoftware.metajnl.com/articles/10.5334/jors.127/">https://openresearchsoftware.metajnl.com/articles/10.5334/jors.127/</a></td>
</tr>
</tbody>
</table>
2. Initial elimination

<table>
<thead>
<tr>
<th>Framework</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dojo Charting</td>
<td>Part of a bigger toolkit</td>
</tr>
<tr>
<td>cytoscape</td>
<td>For visualization of networks only</td>
</tr>
<tr>
<td>chartkick.js</td>
<td>Implementation of chart.js</td>
</tr>
<tr>
<td>Flot Charts</td>
<td>Alpha version</td>
</tr>
<tr>
<td>rickshaw</td>
<td>Website not accessible</td>
</tr>
<tr>
<td>d3.js</td>
<td>No predefined graphs. Need to make/design your own.</td>
</tr>
<tr>
<td>Vaadin Charts</td>
<td>Online tool</td>
</tr>
<tr>
<td>Google Charts</td>
<td>Online tool</td>
</tr>
<tr>
<td>Chart Builder By</td>
<td>Online tool</td>
</tr>
<tr>
<td>Livegap</td>
<td></td>
</tr>
<tr>
<td>cubism.js</td>
<td>Only visualizing time series</td>
</tr>
<tr>
<td>Dygraphs</td>
<td>Only draws line graphs and uses CSV-data.</td>
</tr>
<tr>
<td>n3 charts</td>
<td>Only for AngularJS</td>
</tr>
<tr>
<td>ember charts</td>
<td>Only for Ember.js framework</td>
</tr>
<tr>
<td>Charts 4 PHP</td>
<td>Only for PHP</td>
</tr>
<tr>
<td>recharts</td>
<td>Only for React</td>
</tr>
<tr>
<td>victory</td>
<td>Only for React</td>
</tr>
<tr>
<td>sigmajs</td>
<td>Only network graphs</td>
</tr>
<tr>
<td>YUI Charts</td>
<td>Part of a bigger framework</td>
</tr>
<tr>
<td>gRaphael</td>
<td>Plugin for Raphael which site dont work</td>
</tr>
<tr>
<td>protovis</td>
<td>Predecessor of D3.js</td>
</tr>
<tr>
<td>Factmint Charts</td>
<td>Website not accessible</td>
</tr>
<tr>
<td>smoothie charts</td>
<td>Only for visualizing streaming data</td>
</tr>
<tr>
<td>xcharts</td>
<td>Website in chinese</td>
</tr>
</tbody>
</table>
3. Not free for commercial use

<table>
<thead>
<tr>
<th>Framework</th>
<th>Free for commercial use?</th>
</tr>
</thead>
<tbody>
<tr>
<td>amCharts</td>
<td>Free with link</td>
</tr>
<tr>
<td>AnyChart</td>
<td>no</td>
</tr>
<tr>
<td>CanvasJS</td>
<td>no</td>
</tr>
<tr>
<td>canvasXpress</td>
<td>no</td>
</tr>
<tr>
<td>DevExtreme</td>
<td>no</td>
</tr>
<tr>
<td>DHTMLX Charts</td>
<td>no</td>
</tr>
<tr>
<td>FusionCharts</td>
<td>no</td>
</tr>
<tr>
<td>Highcharts</td>
<td>no</td>
</tr>
<tr>
<td>KoolChart</td>
<td>no</td>
</tr>
<tr>
<td>OLAPCharts</td>
<td>no</td>
</tr>
<tr>
<td>Shield UI</td>
<td>no</td>
</tr>
<tr>
<td>Syncfusion</td>
<td>no</td>
</tr>
<tr>
<td>TeeChart JS</td>
<td>no</td>
</tr>
<tr>
<td>VanCharts</td>
<td>no</td>
</tr>
<tr>
<td>Webix UI</td>
<td>no</td>
</tr>
<tr>
<td>ZoomCharts</td>
<td>no</td>
</tr>
<tr>
<td>ZingChart</td>
<td>with link</td>
</tr>
<tr>
<td>js charts</td>
<td>with watermark</td>
</tr>
</tbody>
</table>
4. Frameworks not available on NPM

<table>
<thead>
<tr>
<th>Framework</th>
<th>NPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimple</td>
<td>no</td>
</tr>
<tr>
<td>EJS Chart</td>
<td>no</td>
</tr>
<tr>
<td>elycharts</td>
<td>no</td>
</tr>
<tr>
<td>jqPlot</td>
<td>no</td>
</tr>
<tr>
<td>jquery sparklines</td>
<td>no</td>
</tr>
<tr>
<td>NextCharts</td>
<td>no</td>
</tr>
<tr>
<td>PlusCharts</td>
<td>no</td>
</tr>
<tr>
<td>RGraph</td>
<td>no</td>
</tr>
</tbody>
</table>

5. Last commit and contributors

<table>
<thead>
<tr>
<th>NPM</th>
<th>Last commit</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>morris.js</td>
<td>2016-09-13</td>
<td>39</td>
</tr>
<tr>
<td>flot</td>
<td>2014-04-29</td>
<td>51</td>
</tr>
<tr>
<td>tui-chart</td>
<td>2018-04-25</td>
<td>9</td>
</tr>
<tr>
<td>flotr2</td>
<td>2014-07-21</td>
<td>19</td>
</tr>
<tr>
<td>uvcharts</td>
<td>2017-01-02</td>
<td>14</td>
</tr>
<tr>
<td>webcharts</td>
<td>2018-03-08</td>
<td>7</td>
</tr>
<tr>
<td>jenscript</td>
<td>2017-11-13</td>
<td>1</td>
</tr>
<tr>
<td>@reactivelib/chart</td>
<td>2018-04-18</td>
<td>1</td>
</tr>
</tbody>
</table>
6. Interview questions and answers

Interview questions and answers in swedish. Interviews conducted with 3 subjects, here named S1, S2, S3 for the sake of anonymity.

Spelar licens någon roll?
S1 - Spelar egentligen ingen roll. Behövs det så köps det in. Det är situationen som avgör.
S2 - Det tar lång tid att köpa in mjukvara/licenser så open-source är bra.
S3 - Börja med open-source. Annars kan det ta tid att få fram och dessutom måste det finnas ett stort värde för att det ska köpas in.

Vart ska det hämtas ifrån? NPM, GitHub eller annat?
S1 - Rekommenderas att det finns att hämta från NPM/Maven eller liknande så det kan integreras i byggregression.
S2 - Spelar inte så stor roll.
S3 - Föredrar NPM då det ingår en viss trygghet.

Behöver det vara kompatibelt med någon mjukvara? Ramverk, webbläsare?
S1 - IE11/Edge + Angular 5. Helst inga egna konsiga dependencies som “silverlight” eller liknande för att det inte ska orsaka kompatibilitetsproblem.
S2 - IE11 och Angular 5.
S3 - IE11 och Angular 5.

Behöver det vara responsivt? Användas på dator och/eller mobil?
S3 - Främst för dator.

Behöver ramverket underhålls regelbundet och uppdateras?
S2 - Viktigt med levande mjukvara. Många användare och att issues fixas.
S3 - Ja. Viktigt att det underhålls och att nya versioner släpps kontinuerligt.

Ska det vara stort och populärt eller spelar det mindre roll?
S1 - Viktigt. Kolla NPM-nedladdningar, GitHub-stjärnor etc.
S3 - Ja. Viktigt att det är stort och har många användare.
Behöver det ha bra dokumentation?

S1 - Bra dokumentation och kodexempel är jätteviktigt för att komma igång. Inte bara API specifikation.
S2 - Bra om det finns tydlig dokumentation och kodexempel.
S3 - Viktigt med bra dokumentation och kodexempel.

Är det positivt ju lättare det är att integrera?

S1 - Ja. Se dokumentation.
S2 - Ja. Se dokumentation.
S3 - Ja. Se dokumentation.

Vilka grafftyper önskas?

S1 - Så det täcker de behov som finns just nu. Basutbud i första hand. Funktion före form.
S2 - Det beror på. Basutbud räcker just nu.
S3 - Bra med flera olika val.

Bör det vara visuellt tilltalande?

S1 - Klart det ska vara snyggt. Lättanvänt är viktigt.
S2 - Design är inte jätteviktigt. Funktionen viktigaste. Dock skadar det inte om det är tilltalande.
S3 - Funktion före form. Bra om det ser bra ut.

Ska det gå att konfigurera utseende, färg, rubriker etc?

S1 - Det får gärna gå att konfigurera så mycket som möjligt.
S2 -
S3 - Bra om man kan ändra designen lätt.

Ska det finnas “mouseover”/tooltips?

S1 - Ja
S2 - Ja
S3 - Ja

Ska man kunna klicka i grafer och fånga events?

S1 - Ja
S2 - Ja
S3 - Ja

Behöver det gå att zooma i grafer?

S1 - Enkelhet före funktion. Hellre enkla funktionella grafer än massa zoom och annat i grafen. Tydligt och enkelt.
S2 - Nej. Inget krav.
S3 - Ja
Något annat?

S1 - Inhousekunskap kan väga tungt. Har något annat team använt liknande som man kan bolla ideer och byta erfarenhet med. Dock svårt att veta vad andra team använder.
S2 - Får gärna finnas inhousekompetens. Det ska vara enkelt och förståeligt och kunna visa data översiktligt och snyggt.
S3 - Viktigt att det är lättanvänd, stort och har många användare, passande funktioner för tillfället. Bra också om det används av andra team inhouse, dock väldigt svårt att veta vad de andra teamen använder.