DALARNA
UNIVERSITY

Evaluation of the Blazor and Angular frameworks performance for web applications

Samuel Nilsson

Supervisor: Vijay Pratap Paidi

Examiner: Somayeh Aghanavesi

Thesis for Bachelor Degree in Informatics (GIK28T)

Dalarna University, Sweden – www.du.se

ABSTRACT

Introduction

Blazor is a new framework and current research show that there is a lack of performance comparisons. Therefore, a need to compare and evaluate this new Blazor framework is ought to be requested to show if it can compete with one of the best and most used frameworks, Angular. It is also one of the few web developing frameworks that has moved away from using JavaScript and therefore a comparison with a framework that uses JavaScript will be very interesting.

Aim

The aim of this study is to make a performance comparison between two web applications using the Angular and Blazor frameworks. I will evaluate the performance and present the results to see if Blazor can compete with Angular as a modern web development framework. More specifically, the study aims to answer the following research question:

Based on a performance evaluation would it be favorable for a developer or company to consider using the Blazor framework over the Angular framework when developing web applications?

Method

I started by doing a literature review of the field I wanted to research. I used du.se library search engine, which had access to the DiVa portal, Google scholar, Summon and Libris etc. The literature was found using the keywords "Blazor", "Angular", "Web application", "Framework", "Comparison" and "Evaluation". Thereafter, two web applications were developed for the purpose of performance evaluation of the two frameworks. An evaluation tool was used to create the report based on the measurement criterions: "First Contentful Paint", "Time to Interactive", "Speed Index", "Total Blocking Time", "Largest Contentful Paint" and "Cumulative Layout Shift". When evaluating I summed up which framework that was more favorable in each category and displayed it in tables and graphs.

Results

I found that the Angular framework was more favourable in 4 out of the 6 criterions. Even though Angular performed better overall it showed some very unstable performance results during the twenty tests. Blazor displayed a stable performance throughout all twenty tests in all categories and was very fast and responsive once the initial load of the client was done.

Conclusions

Based on the results Angular was the more favourable framework during my tests, due to overall performance. But the Angular framework showed too many inconsistencies during the tests to call it superior, meanwhile Blazor performed very stable results. This indicates that the Blazor framework can possibly compete with the Angular framework as a modern web development framework. Thus, my conclusion is that it would be favourable for developers and companies to consider using the Blazor framework for new projects, especially if they have previous experience with the C# language or are new to web developement.

Keywords Blazor, Angular, Framework, Web application, Evaluation, Comparison

TABLE OF CONTENTS

1 Introduction

Since the World Wide Web was introduced to the world, a very hot topic was to make the static markup language HTML more dynamic and interactive. This was finally achieved during the middle of the 90s with the introduction of the client-side scripting language, JavaScript (Sandberg, 2021). JavaScript became quickly one of the most popular scripting languages when developing web applications. This led Google to inventing the web development framework, Angular, in September 2016 (Angular, 2021). However, lately new languages have become available when developing dynamic and interactive web applications. Microsoft released a new web development framework, Blazor, in May 2020, which is based on basic HTML markup language mixed with C# to achieve the coveted dynamic behavior on a web application (Blazor, 2021).

Blazor is a new framework and current research show that there is a lack of performance comparisons. Therefore, a need to compare and evaluate this new Blazor framework is ought to be requested to show if it can compete with one of the best and most used frameworks, Angular. It is also one of the few web developing frameworks that has moved away from using JavaScript and therefore a comparison with a JavaScript using framework will be very interesting.

To develop modern front-end web applications, do developers really need to rely on JavaScript using frameworks? Kozak & Smołka (2020) believed that despite the constant development of the ASP.NET (Active Server Pages) platform, developers are forced to create their front-end using frameworks that use JavaScript language, i.e., Angular. This was until .NET Core 3.0 presented the Blazor framework on May 19th, 2020. This allowed developers to create single-page apps (SPA) using C# mixed with traditional HTML without the need of JavaScript language (Kozak & Smołka, 2020). Can Blazor keep up with one of the leading JavaScript using web development frameworks when we evaluate the performance?

Ever since the introduction of the JavaScript language, new scripting languages whose purpose was to run on web servers were invented, such as PHP and Python. According to Sandberg (2020) this meant that software libraires were expected to run several external tasks and thus the first web frameworks emerged. Frameworks are usually a set of libraries and tools which main goal is to aid the developer during the process of building a web application. Usually allowing the developer to begin programming right away without the need of finding suitable software technologies (Kozak & Smołka, 2020).

My purpose with this study is to evaluate if Microsoft's web framework Blazor can compete with one of the leading web development frameworks, Angular. To do this I have created two minor web applications using Blazor for the first app and Angular for the second app. To make this evaluation as even-handed as possible I have used the same backend (server side) code for both applications. I have also used the same basic operations Create, Read, Update and Delete (CRUD) to read and modify data from the backend. This means that the only thing that differentiate these apps from each other is the frontend part which uses the different frameworks. To measure the performance, I have used “Lighthouse” which is an open-source, automated tool created by Google (Tools for web developers, 2021). By doing this my evaluation will show if it is favorable for web developers or companies that want to strive away from JavaScript to consider using Blazor instead.

1.1 Aim

The aim of this study is to make a performance comparison between two web applications using the Angular and Blazor frameworks. I will evaluate the performance and present the results to see if Blazor can compete with Angular as a modern web development framework. More specifically, the study aims to answer the following research question:

- Based on a performance evaluation would it be favorable for a developer or company to consider using the Blazor framework over the Angular framework when developing web applications?

2 Theory/Literature review

There is a lot of information and guides on the internet in relation to creating web applications with the Blazor framework, however there is not much to find regarding studies and comparisons of the new framework. A factor might be that it is only two years old by the time this thesis was written. Therefore, a comparison in performance and usability with one of the leading JavaScript frameworks, Angular, would be beneficial for this relatively new framework.

Sanberg, E (2020) did an evaluation of the Blazor framework where he compared it briefly with the most common front-end frameworks like Vue, Angular, Ember and React. The evaluation criteria used were documentation, lines of code, community size, framework usage, maturity, freshness, browser support and framework cost. During the said study Blazor came out on top in six out of eight criteria and that was unexpected by the author. There was also a comparison between the programming languages C# and JavaScript/TypeScript. This was done because it could have an impact on what framework to choose since developers want efficiency, simplicity, and readability (Sandberg, 2020). During research of previous studies and reading through web developer forums on the Internet an interesting and common topic have been the comparisons between the languages JavaScript and C#. This is an interesting field for further studies in the future. That is, why do developers want to search for another front-end programming language than JavaScript?

2.1 Angular

As mentioned earlier the Google team created and introduced the web developing framework Angular back in 2016 (Angular, 2021). Angular is a platform used for web development and is built using TypeScript. Angular is strongly recommending the use of TypeScript over JavaScript nowadays, due to the many central problems known to JavaScript.

The Angular framework is scalable, which means that you can develop small single applications all the way up to enterprise level applications (Angular, 2021). Angular makes heavy use of components as these are the building blocks in an Angular application. For example, a developer can build a header component, a navbar component, a container component, and a footer component and together they combine into a web application. These components' purpose is to offer a strong encapsulation and spontaneous structure of the application. It also makes it easier for developers to read and understand the structure of your code (Angular, 2021). Angular also supports a Command-Line Interface (CLI) tool which allows developers to initialize, develop and maintain the application directly from a command interpreter. For example, when a developer wants to create a new project, they can just type "ng new app" in the command shell. This result in the framework auto generating a new project, with routing support included, to make it easy for the developer to start programming right away. Afterwards they can use the "ng generate" command to auto generate new files for components and services used for the applications needs (Angular, 2021).

2.1.1 JavaScript & TypeScript

When Netscape first introduced JavaScript to the world in 1995, some developers rejected it as a "toy language" and said it was not worth their attention. According to Atwood (2007) this was mainly due to some misstatements about JavaScript by Netscape to avoid position JavaScript as a competitor to Java. But JavaScript has come a long way since then and by the turn of the millennium JavaScript have been accepted as the lingua franca (standardized programming language) for developing web applications (Wang et al., 2019). JavaScript is for the most part compiled by the web browser when certain functions are required and thus achieves both faster execution and faster loading speeds (Kristensen & Møller, 2017). But JavaScript has its own drawbacks, and this led to Microsoft introducing the superset of JavaScript in 2012, called TypeScript. Improving its syntax, adding numerous features such as support for variable types and compile-time type checking, interfaces, enums, namespaces and enhancing the development process. Since it is a superset of JavaScript it means that TypeScript also supports all features that JavaScript does (Kristensen & Møller, 2017).

2.2 Blazor

Blazor is an open-source .NET platform web framework developed by Microsoft. Blazor utilizes a mix of C# and HTML to build the web applications and consists of a server side and a client side where the client side is based on the WebAssembly standards (Blazor, 2021). Similar to Angular, Blazor is also using components and services to create the web applications. The difference is the Razor syntax which allows the developers to use C# and HTML languages. This syntax also allows the C# code to contact JavaScript APIs so the developer can continue to use the large JavaScript libraries that exists while still writing the logic in C# (Blazor, 2021). According to Sandberg (2020) the initial load of a Blazor WebAssembly application will be time consuming, this is due to the Client side of Blazor sending a stripped-down version of the .NET runtime. This stripped-down version will be able to run as WASM on the client when loaded initially and this might affect the first performance evaluation. But once the application has loaded it will lead to fast response times and low need for messages to the server.

2.2.1 C#

The C# language is heavily used when developing Blazor web applications. The programming language C# origins from the classic programming language C and has a C-style syntax, but C# have evolved as a programming language and thus contains a lot more features and possibilities than C

(Skansholm, 2008). Microsoft developed the C# language as a part of the .NET framework in 2002. C# is platform dependent; it means that if a program is written in C#, it can be executed on any platform with a .NET implementation, without the need to change the program. This gives C# great portability features and makes C# a very powerful language to use when developing web applications (Skansholm, 2008).

2.2.2 WebAssembly

WebAssembly or otherwise known as WASM is a binary-instruction format that is designed to perform better than JavaScript. It is designed to allow non-web programming languages such as C/C++, Python and Rust to be used in web development. The idea is to compile the source code to a binary format instead of compiling it to JavaScript, comparable to how C# is usually executed and thus improving the performance (Apell, 2020).

2.3 Lighthouse

Lighthouse is an open-source, automated tool used for improving the quality of web pages. Developers can run it against any website page, public and even if it requires authentication. It is a multipurpose tool with which a developer can test the performance, accessibility, progressive web applications, SEO and that is only the tip of the iceberg (Tools for web developers, 2021).

To run Lighthouse the developer must navigate to it in Chrome DevTools, or from the command line, or as a Node module. Lighthouse creates a report after it has run all the tests so the developer can see how well a web page performed. After the test is performed the developer can read the report where Lighthouse explains why the test result is important as well as how to improve or fix the web page (Tools for web developers, 2021).

3 Metod | Method

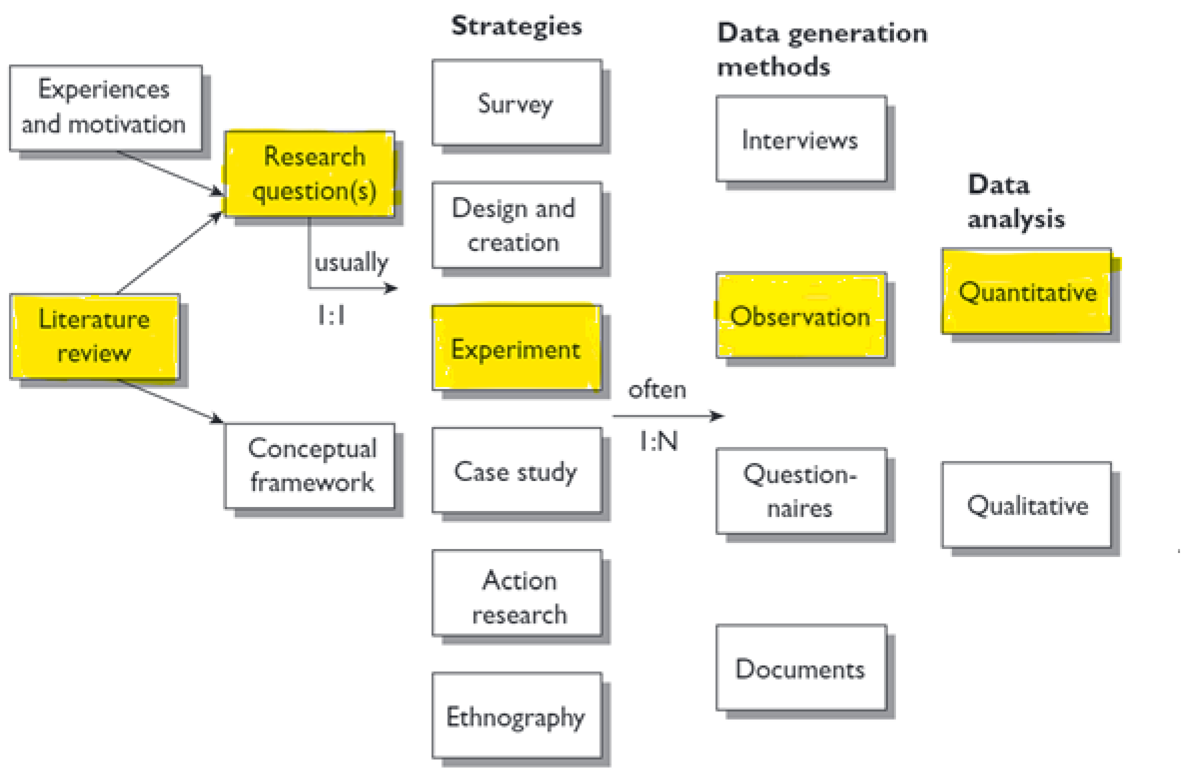


Figure 1 - Quantity method flowchart, first phase includes Literature review and Research question(s), Second phase contains the Experiment, and the third phase describes the observation. By following these steps the data analysis is considered to be quantitative.

3.1 First phase

First, a literature review of the field I wanted to research was performed. I used du.se library search engine, which had access to the DiVa portal, Google scholar, Summon and Libris etc. The literature was found using the keywords “Blazor”, “Angular”, “Web application”, “Framework”, “Comparison” and “Evaluation” to determine what have already been researched and from there the research questions were created. For this thesis, a quantitative method was used to execute the evaluation and comparison. The quantitative method was used when executing the evaluations with help of the evaluation tool, Lighthouse, which created a report with the measurement values.

3.2 Second phase

During the second phase, two separate customer register web applications were developed for Transportstyrelsen (Swedish Transport Agency) with the purpose to measure and evaluate the web applications framework performance. The code editor used to develop and run these applications from was Microsoft's Visual Studio 2019 IDE. For the first web application Blazor WebAssembly framework and .NET Core 3.1 platform were used, and for the second web application the Angular framework along with the .NET Core MVC (Model View Controller) platform were used. My collaboration with Transportstyrelsen had an impact on the comparison and evaluation of these two specific frameworks and the selections of specifications such as platforms, database selection and functions of the application. The applications are connected to the same SQL server database and the applications main functions are to display and edit data using basic CRUD (Create, Read, Update, Delete) operations. I have no previous experience with any of the mentioned frameworks, and I have never programmed with the C# language before, nor TypeScript. This allowed me to explore tutorials, documentation, and community sites from both frameworks and programming languages. I tried to reuse the code and patterns for both web applications as long as it did not harm the quality of the code. This means that the only part that differs between these two web applications is the frontend code and the use of framework template which will allow for a fairer evaluation comparison.

3.3 Third phase

In the third phase I evaluated and compared the two web applications that had been developed. To make this evaluation as fair as possible I used the same software to develop both web applications (Microsoft Visual Studio), the same database (SQLEXPRESS Server), the same web browser (Google Chrome) and the same evaluation tool (Lighthouse). According to Love (2018) Lighthouse is an easy-to-use tool when measuring and evaluating performances of a web applications and it generates a detailed performance report. Therefore, I made the choice to use Lighthouse for my evaluation and comparison of my web applications. Lighthouse has six metrics specified for evaluating performance and thus I chose to include all six metrics for my research. The six-performance metrics I evaluated with Lighthouse are:

- **First Contentful Paint**
  Measures how long it takes for the browser to render the first piece of DOM (Document Object Model) content, for example texts and/or images, after a user navigates to the certain page (Web.dev, 2021).

- **Time to Interactive**
  According to Web.dev (2021) it measures the time it takes for the page to become fully interactive. A page is considered fully interactive when:

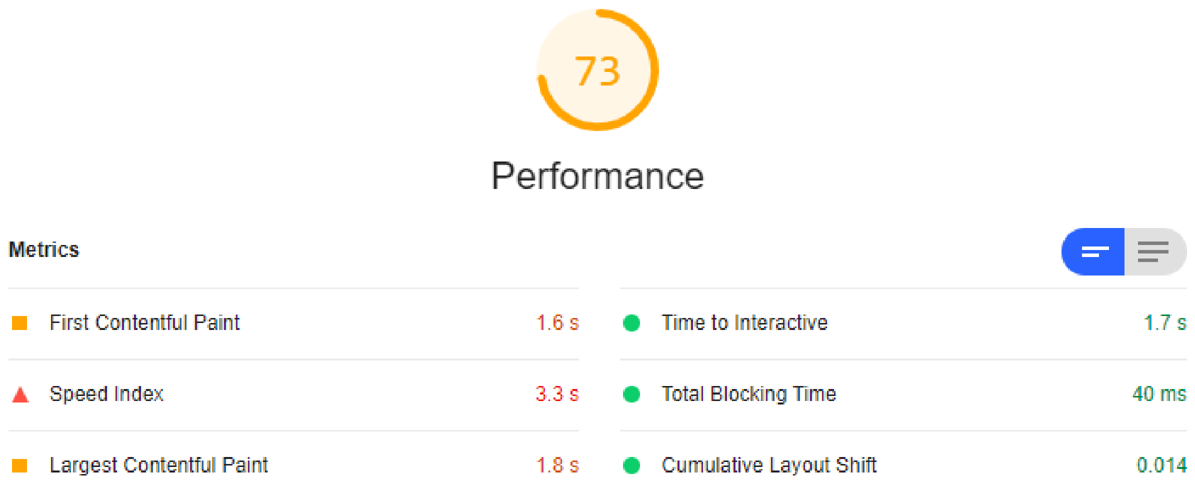
- The page displays useful content, which is measured by the First Contentful Paint.
- Event handlers are registered for most visible page elements.
- The page responds to user interactions within 50 milliseconds.

- **Speed Index**
  Shows how quickly the contents of a page are visually displayed during page load. This is done by capturing a video of the page loading and then measuring the visual progression in seconds between each frame in the video (Web.dev, 2021).

- **Largest Contentful Paint**
  Marks the time at which the largest text or image is rendered to the screen. This approximates when the main content of the page is visible to users (Web.dev, 2021).

- **Total Blocking Time**
  Measures the total amount of time that a page is blocked from responding to user input, such as mouse clicks, screen taps, or keyboard presses (Web.dev, 2021).

- **Cumulative Layout Shift**
  Measures unexpected movement of page content. A layout shift occurs any time a visible element changes its position from one rendered frame to the next (Web.dev, 2021).



Metrics

■	First Contentful Paint	1.6 s	●	Time to Interactive	1.7 s
▲	Speed Index	3.3 s	●	Total Blocking Time	40 ms
■	Largest Contentful Paint	1.8 s	●	Cumulative Layout Shift	0.014

Figure 2 - Example of how a Lighthouse performance report could look like with the six evaluation metrics.

Twenty performance tests were measured with each framework and the average of each metric measurement was calculated and displayed in a chart and a table, which all will be presented later in the result part of this thesis. The differences between the C# and JavaScript/TypeScript programming languages will be discussed since it is relevant for developers or companies when they considering what framework to choose for web development.

4 Results and Discussion

4.1 Results

In this chapter, I will present all my results and findings from the twenty web application framework tests. The data is collected from the Lighthouse reports and inserted into an Excel table to present them as graphs. I have compared every measurement column from each framework and presented all the twenty test results in a graph with a follow-up graph where I present the average result of each column.

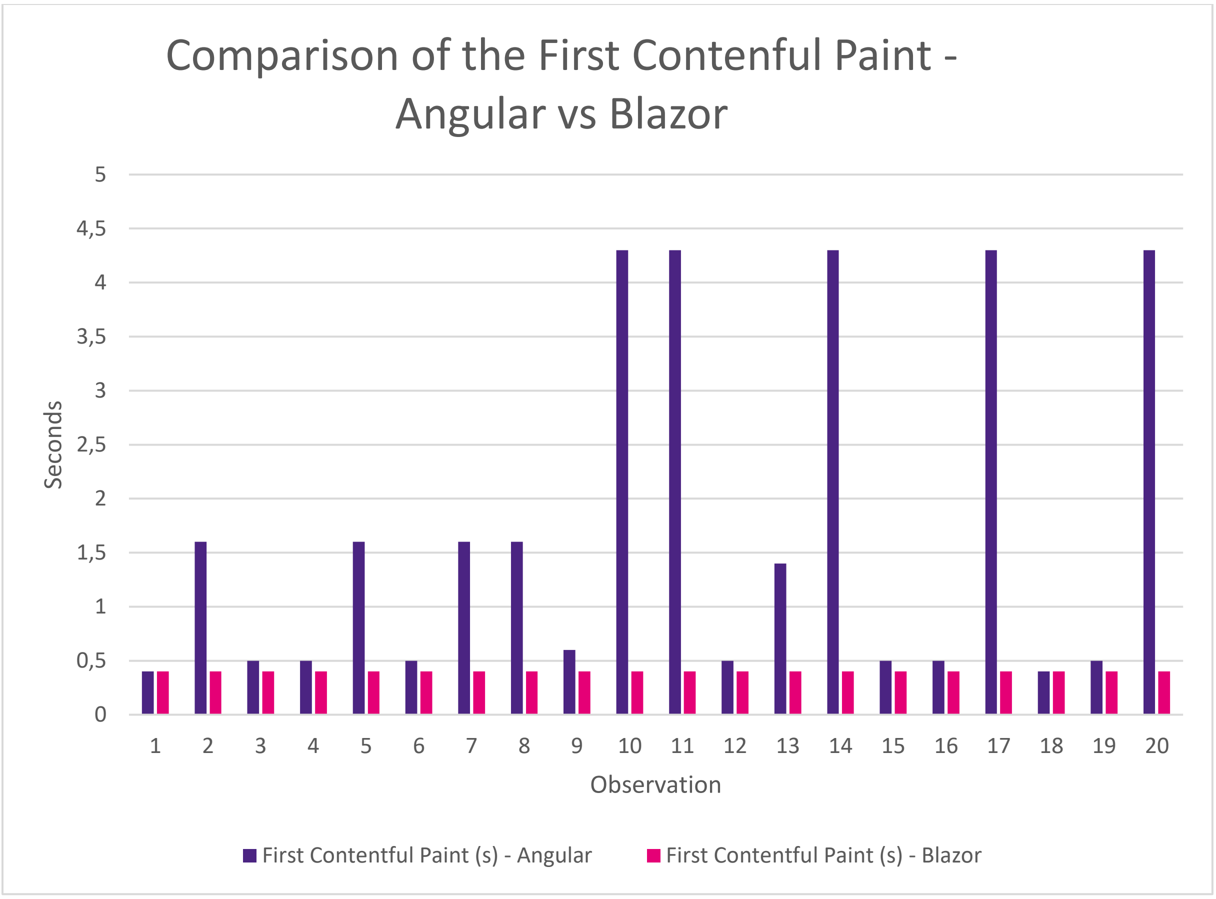


Figure 3 - Comparison of the First Contentful Paint - Angular vs Blazor

Figure 3 shows an observation of the first performance metric, First Contentful Paint, which measures in seconds how long it takes for the web application to show the first text or image when a user navigates to the page (Web.dev, 2021). It is notable that during the twenty observations, Blazor showed a stable performance of 0.4 seconds while Angular showed a varying performance ranging from 0.4 and all the way up to 4.3 seconds.

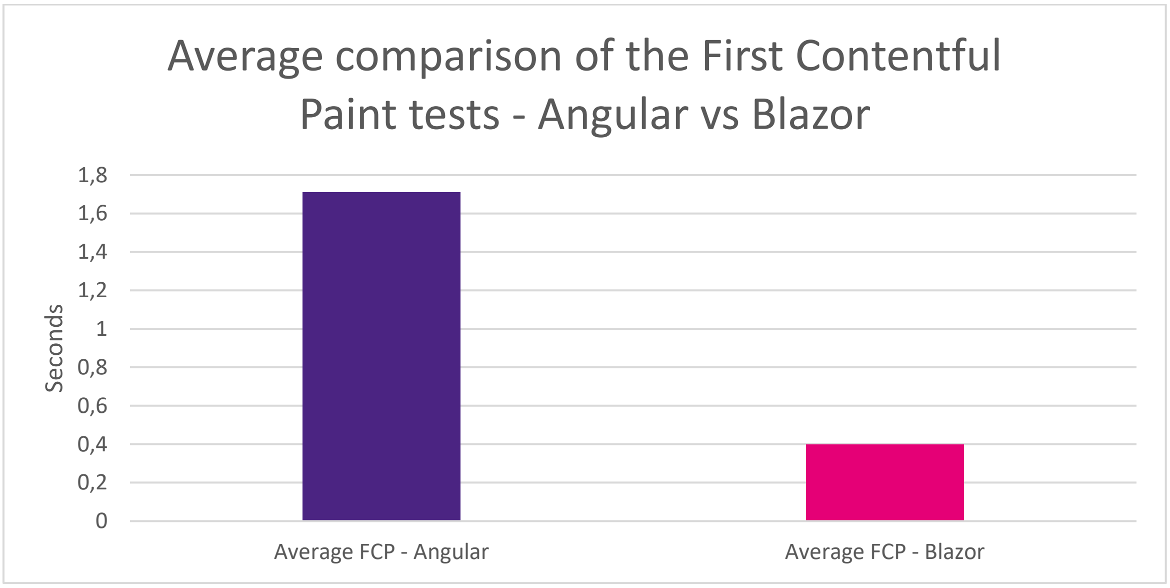


Figure 4 - Average comparison of the First Contentful Paint - Angular vs Blazor

Figure 4 presents the average calculations of the twenty observations for each framework. Since Blazor had a very stable performance they ended up on an average of 0.4 seconds to load the first text or image. Angular was over a second slower to load the first text or image with an average of 1.7 seconds.

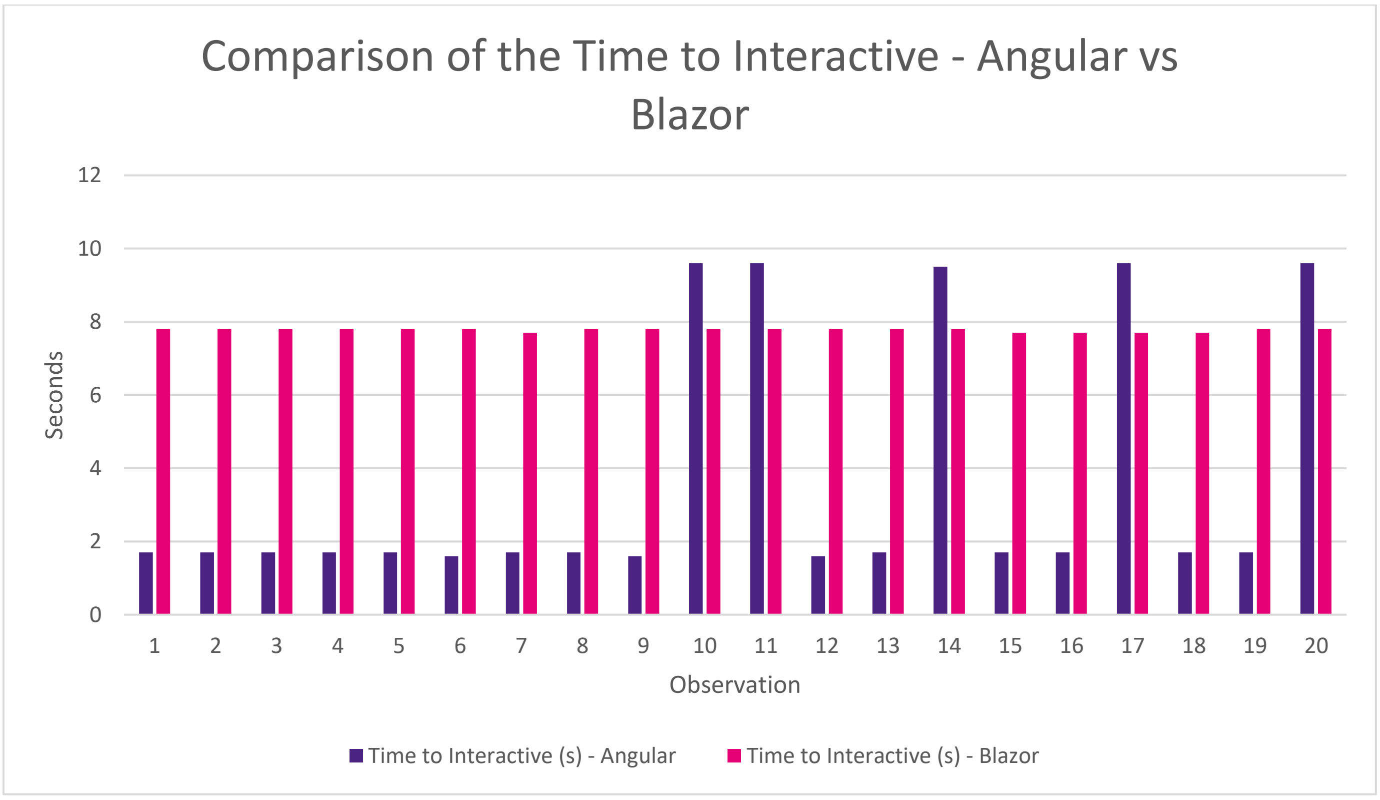


Figure 5 - Comparison of the Time to Interactive - Angular vs Blazor

Figure 5 displays the observation of the second performance metric, Time to Interactive, which measures how many seconds it takes for the page to become fully interactive (Web.dev, 2021). It is again notable that Blazor shows a very stable performance of 7.8 seconds, meanwhile Angular varies in the "Time to Interactive" category from 1.7 and all the way up to 9.6 seconds.

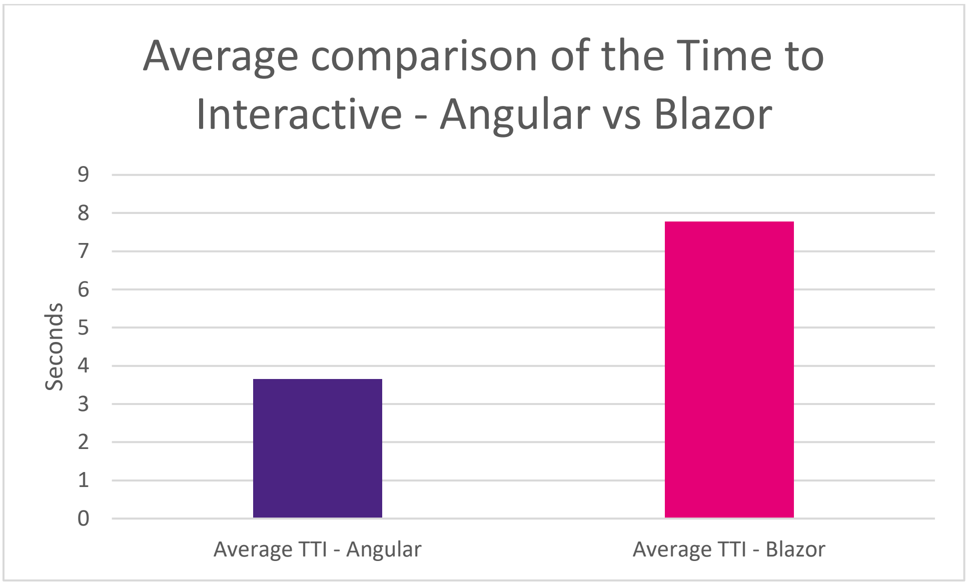


Figure 6 - Average comparison of the Time to Interactive - Angular vs Blazor

Figure 6 shows us the average calculations of the twenty observations for each framework. The graph shows that despite Angulars irregular performance they ended up with an average of 3.65 seconds in the "Time to Interactive" category while Blazor measured 7.75 seconds.

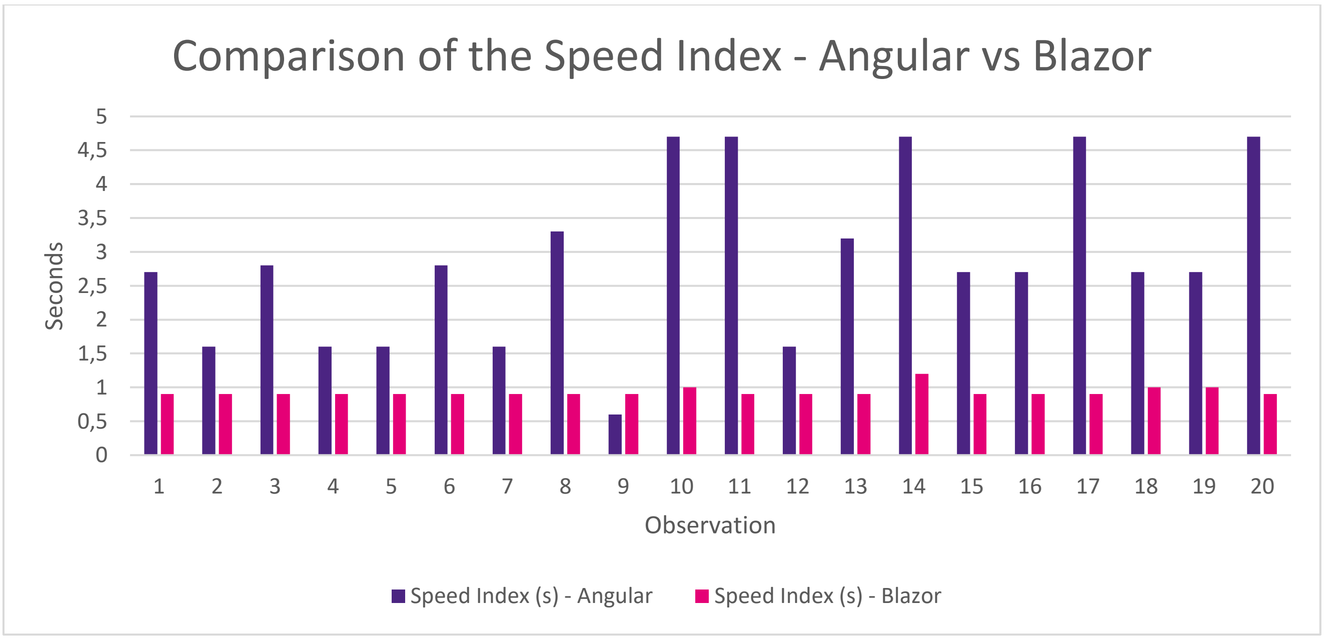


Figure 7 - Comparison of the Speed Index - Angular vs Blazor

Figure 7 displays the observation of the third performance metric, Speed Index, which measures in seconds how fast the contents of a page are presented for the user during page load (Web.dev, 2021). Once again Blazor presented a very reliable result ranging from 0.9-1.2 seconds, Angular showed a very unstable result ranging from 0.6-4.7 seconds.

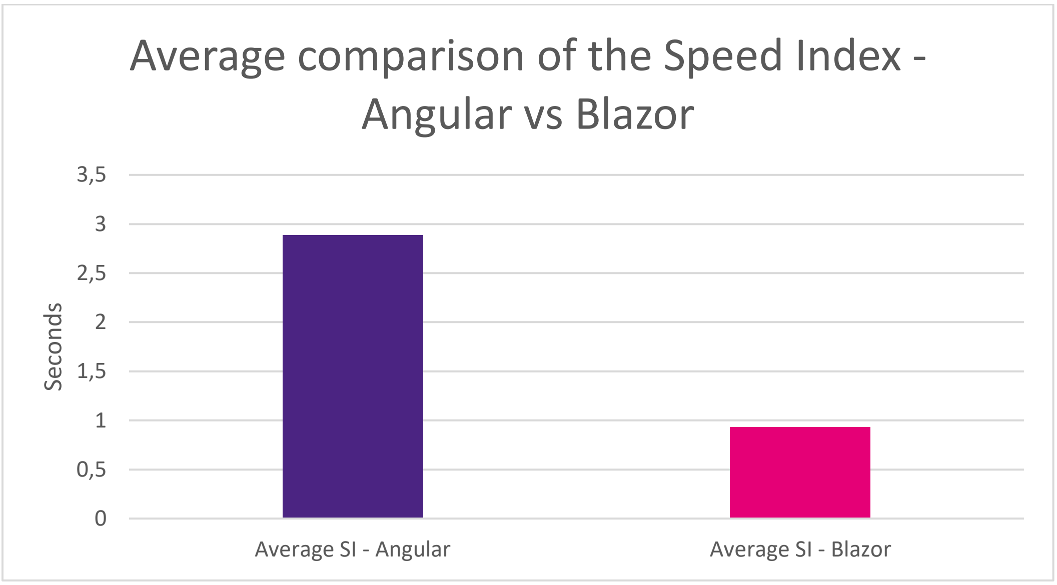


Figure 8 - Average comparison of the Speed Index - Angular vs Blazor

Figure 8 demonstrates the average calculations of the twenty observations for each framework. Blazor reliable performance managed to get an average Speed Index of 0.93 seconds meanwhile Angular showed a very unstable performance and managed to average a Speed Index of 2.88 seconds.

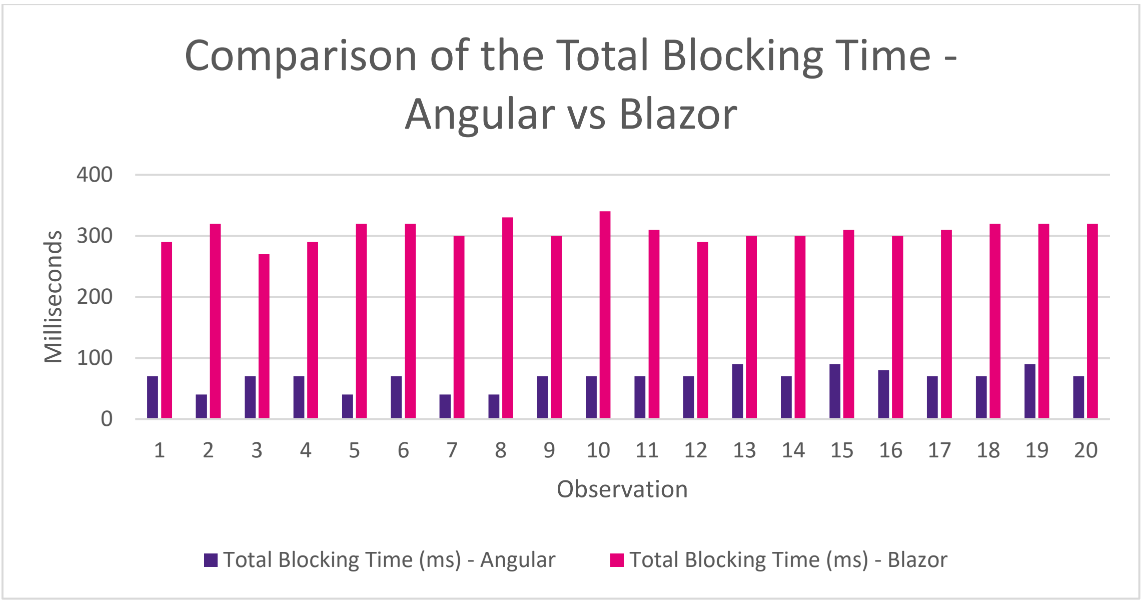


Figure 9 - Comparison of the Total Blocking Time - Angular vs Blazor

Figure 9 presents the observation of the fourth performance metric, Total Blocking Time, and it measures the total amount of time in milliseconds that a page is blocked from responding to user input such as mouseclicks, keyboard inputs and screen taps for example (Web.dev, 2021). Both frameworks performed very stable in this category, Blazor ranged from 270-340 milliseconds and Angular ranged from 40-90 milliseconds of "Total Blocking Time".

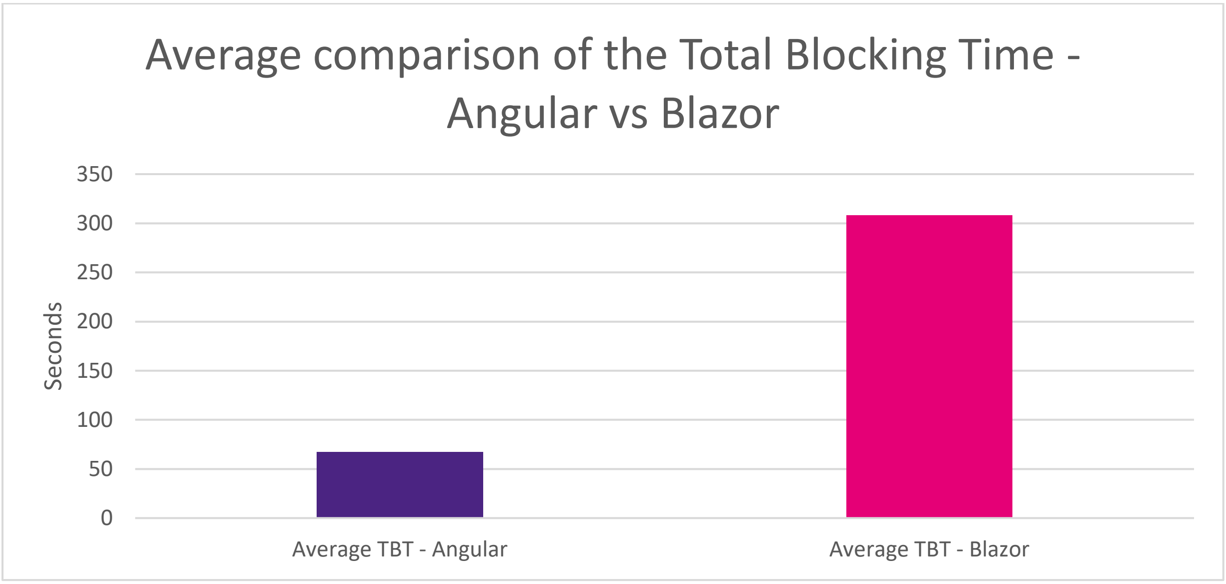


Figure 10 - Average comparison of the Total Blocking Time - Angular vs Blazor

Figure 10 shows the average calculations of the twenty observations for each framework. In the "Total Blocking Time" category both frameworks had a very reliable performance, Angular performed a value of 67.5 milliseconds and Blazor 308 milliseconds.

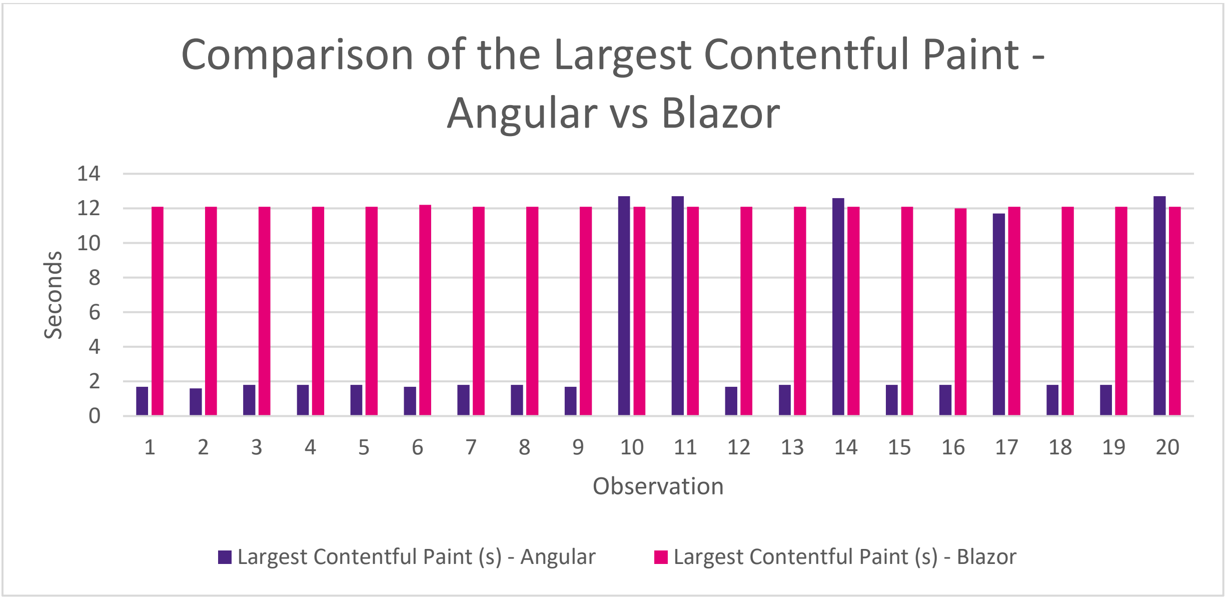


Figure 11 - Comparison of the Largest Contentful Paint - Angular vs Blazor

Figure 11 displays the observation of the fifth performance metric, Largest Contentful Paint, which measures in seconds at which the largest text or image is painted for the user (Web.dev, 2021). Blazor is once again displayed a solid result ranging from 12-12.2 seconds, Angular on the other hand ranged from 1.6-12.7 seconds during the twenty tests.

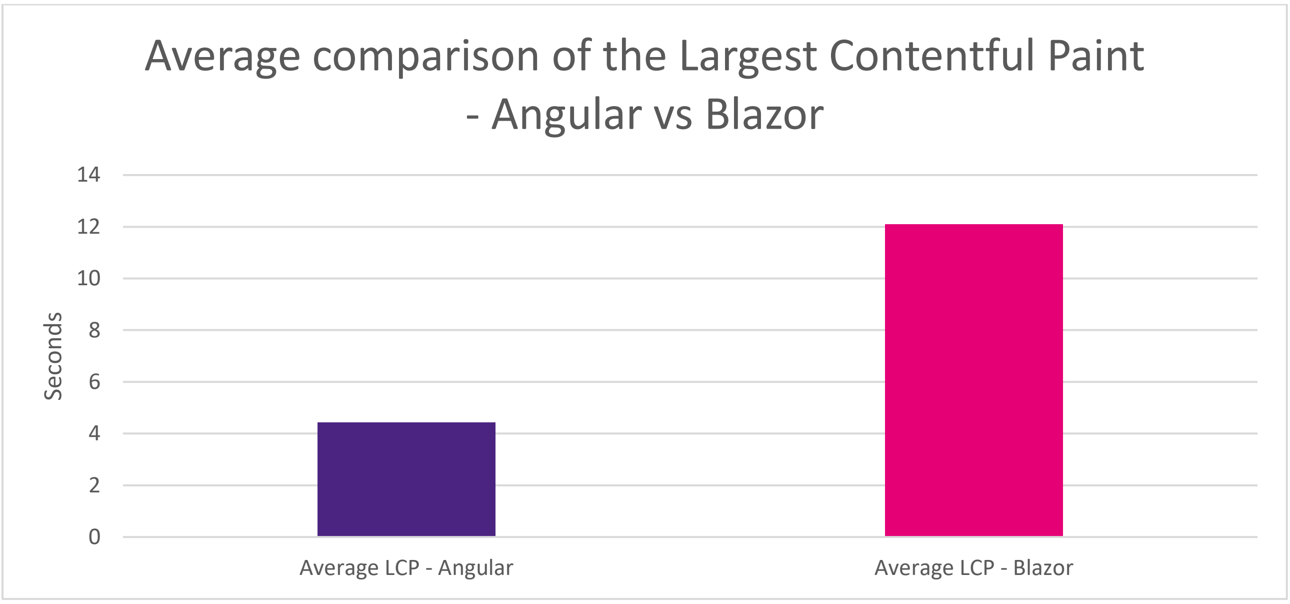


Figure 12 - Average comparison of the Largest Contentful Paint - Angular vs Blazor

Figure 12 is displaying the average calculations of the twenty observations for each framework. Even though Angular was unreliable in its performance it still performed better than Blazor in the "Largest Contentful Paint" category. Angular presented an average result of 4.44 seconds and Blazor averaged 12.1 seconds.

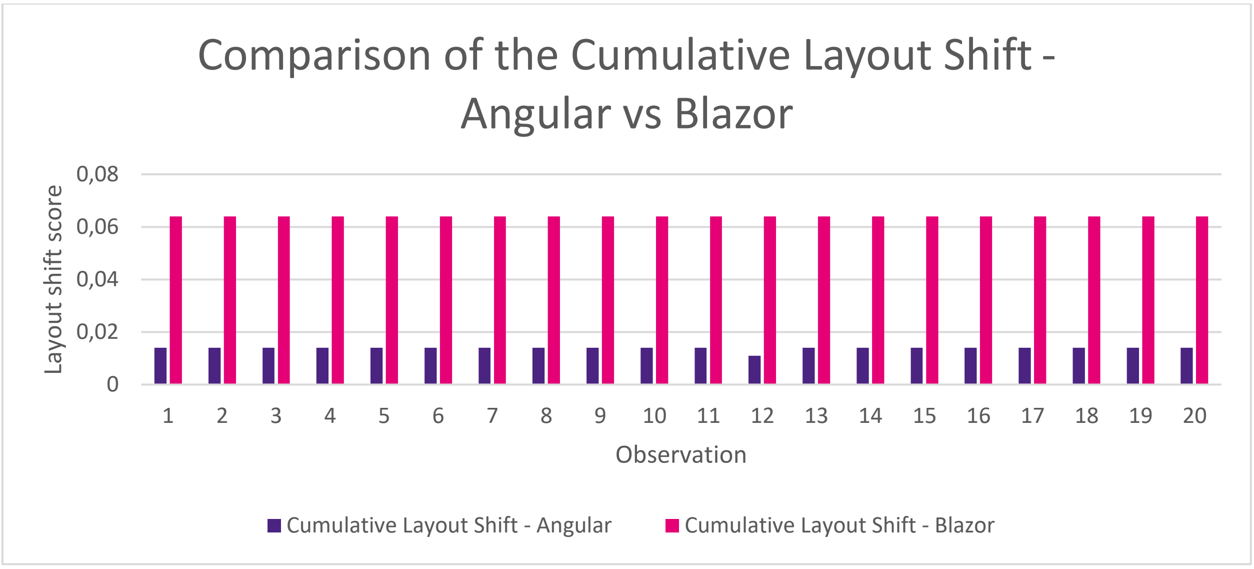


Figure 13 - Comparison of the Cumulative Layout Shift - Angular vs Blazor

Figure 13 displays the observation of the sixth performance metric, Cumulative Layout Shift, which measures how links, images, and text moves on the page from one frame to the next frame, this is to provide a good user experience. According to the website Web.dev (2021) a score under 0.1 is considered good, 0.1-0.25 means it needs improvement and everything above 0.25 is considered bad for the user experience. Both frameworks performed a score under the 0.1 threshold, Angular scored a value of 0.014 and Blazor got a value of 0.064.

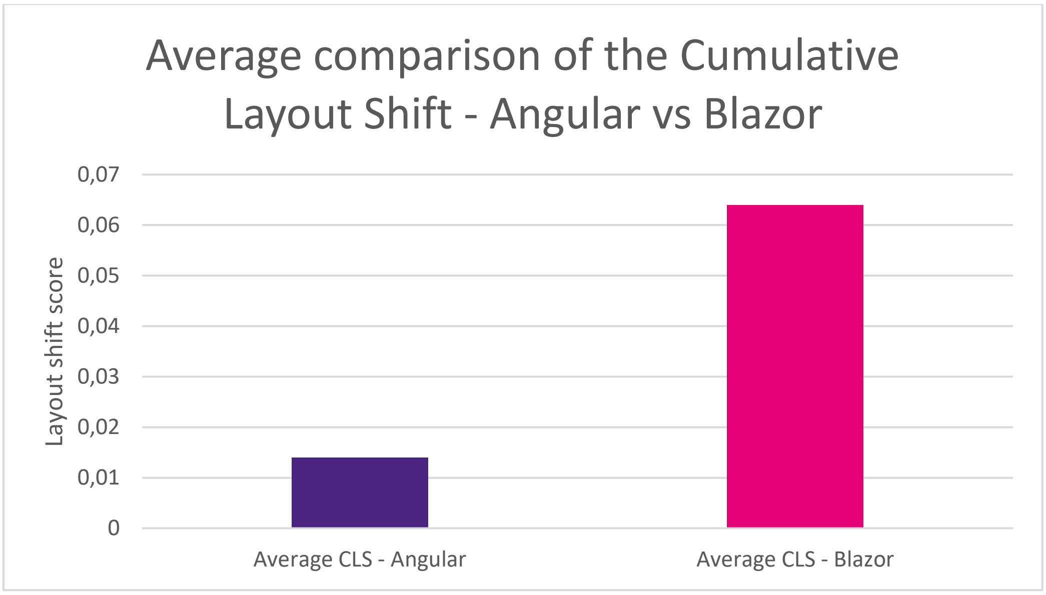


Figure 14 - Average comparison of the Cumulative Layout Shift - Angular vs Blazor

Figure 14 demonstrates the average calculations of the twenty observations for each framework. Since both frameworks performed very reliable during these twenty tests the average did not change for neither of the frameworks. Angular scored an average of 0.014 and Blazor scored and average of 0.064.

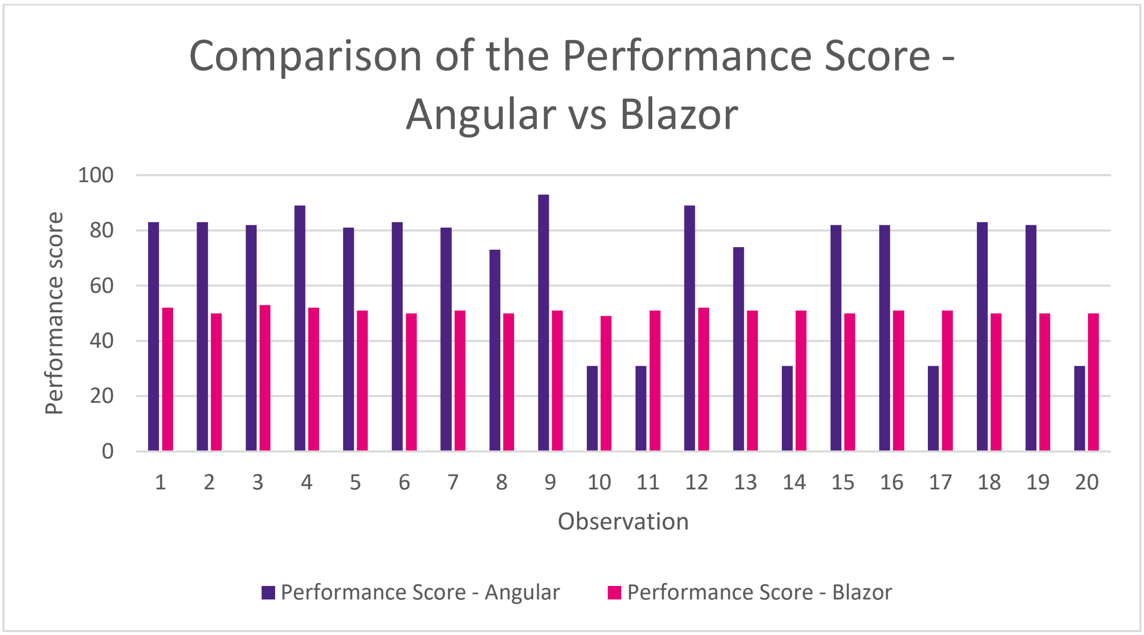


Figure 15 - Comparison of the Performance Score - Angular vs Blazor

Figure 15 will display the total Performance Score of all six categories during the twenty tests. This was calculated by the software Lighthouse and presented in its report, the higher the score the better was the performance. It is notable that Blazor had a steady Performance Score with a small range from 49-53 and Angular ranged all the way from 31-93.

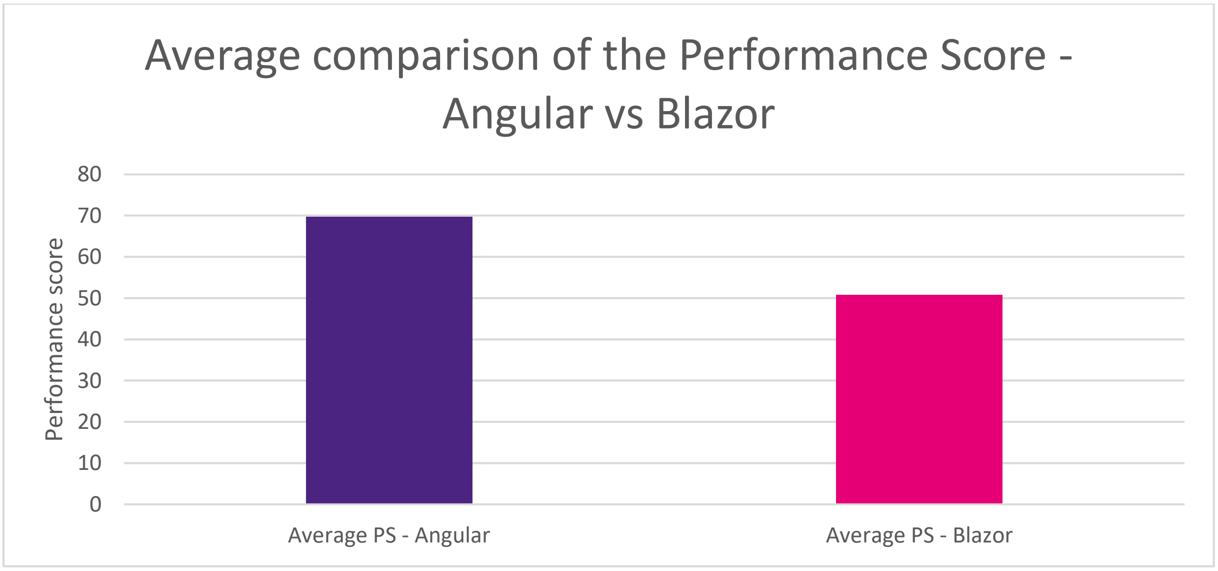
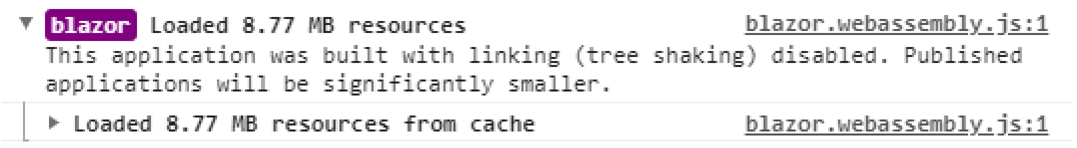


Figure 16 - Average comparison of the Performance Score - Angular vs Blazor

Figure 16 will present the average Performance Score results. Blazor scored an average of 50.8 and Angular scored an average of 69.75 due to the unstable performance values.

4.2 Discussion

The aim of this thesis was to compare and evaluate the Blazor framework with the Angular framework to see if it could compete as a modern web development framework. Hence, did Blazor perform on the same level as the Angular framework? Even though the Angular frameworks total Performance Score result was better than the Blazor framework (Angular scored 69.75 and Blazor 50.8) it would be unfair to say that Blazor performed worse than Angular. This is mainly due to the Angular framework applications worrying pattern of unreliable performance in four out of six measured categories. Meanwhile the Blazor framework application had a very stable and reliable performance throughout each category and even outperformed Angular in two out of six categories by a huge margin (First Contentful Paint and Speed Index) which was surprising to see. It is also worth noting that these tests were run locally from my computer at home since I had neither time nor resources to fully publish these applications on a live http server. This might have an impact on especially the Blazor framework application since it is built with the WebAssembly platform. Therefore, it must download a considerable amount of data to the client the first time a user navigates to the page as shown by figure 17.



```
▼ blazor Loaded 8.77 MB resources                          blazor.webassembly.js:1
  This application was built with linking (tree shaking) disabled. Published
  applications will be significantly smaller.
  ▶ Loaded 8.77 MB resources from cache                    blazor.webassembly.js:1
```

Figure 17 - Picture from the console after I have started the Blazor framework WebAssembly application.

It is difficult to say how much it would affect the Performance Score for Blazor to publish the application, but it would be an interesting project for future research.

The Angular framework did very well in four out of the six performance measurements (Time to Interactive, Total Blocking Time, Largest Contentful Paint, Cumulative Layout Shift) even though it had some troubles performing stable results. This data suggest that the Angular framework is performing better than the Blazor framework using these performance metrics. But I do not want to diminish the Blazor frameworks performance as I must keep in mind that it could be improved and maybe even surpass the Angular framework if the application was published.

Another factor that might impact the selection of framework for a developer or a company is the programming language that is used for the code's logic. Since the programming language is one of the main differences between these frameworks it is an interesting point to discuss. Blazor uses C# for its logic and Angular uses JavaScript/TypeScript for its logic. Sandberg (2020) stated in his thesis report that one of benefits with C# is its broad user base. As of May 2021, C# is ranked among the five most popular programming languages among developers, nearly twice as popular as JavaScript that is placed at rank 7 (Tiobe.com, 2021). This simplifies when a company wants to hire new developers as there will be more developers to choose from with experience of C# than JavaScript. The learning curve for a developer, or a team, moving to web development will not be that steep if they can write the web applications logic in a programming language that they are already familiar with.

When taking these factors into account, Blazor might have an advantage over Angular because of the C# language. This is specifically true if the user has previous experience with C# or another comparable language to it, like C or C++, which are both among the five most popular programming languages (Tiobe.com, 2021).

5 Conclusions

Out of the six performance criteria measured, the evaluation favored Angular in four cases. Table 1 will demonstrate the results in a clearer view and displaying which framework was favored in which category.

Criterion	Framework Evaluation favors
First Contentful Paint	Blazor
Time to Interactive	Angular
Speed Index	Blazor
Total Blocking Time	Angular
Largest Contentful Paint	Angular
Cumulative Layout Shift	Angular
Performance Score	Angular

Table 1 - The final results from the tests

Based on the evaluation results from this thesis report, Angular performs better than Blazor. However, even though Angular performed better than Blazor it does not mean that it is necessarily more superior as a framework. Angular showed too many inconsistent results during the evaluation tests for it to be superior, meanwhile Blazor produced very consistent performance results and seemed like a stable framework overall during the test phase. As stated in the discussion the Blazor framework's performance could be improved and maybe even surpass the Angular framework if the application was published. This indicates that the Blazor framework can possibly compete with the Angular framework as a modern web development framework.

Even though Blazor performed worse than Angular during the evaluation, it would be favorable for developers and companies to consider it as a framework for new projects. Especially if the user is new to web development or has previous experience with C# or another comparable programming language.

6 Reference List

Articles

Apell, M. (2020). Developing Blazor Web Application Running on Microsoft Azure. [Bachelor Thesis, Metropolia]. Retrieved from https://www.theseus.fi/bitstream/handle/10024/335341/Apell_Mika.pdf?sequence=2

Freeman (2019). Essential Angular for ASP.NET Core MVC 3 [Elektronisk resurs]. Apress.

Kozak, K., & Smołka, J. (2020). Analysis of the Blazor framework in client-hosted mode. Journal of Computer Sciences Institute, 16, 269-273. https://doi.org/10.35784/jcsi.2019

Kristensen E.K., Møller A. (2017) Inference and Evolution of TypeScript Declaration Files. In: Huisman M., Rubin J. (eds) Fundamental Approaches to Software Engineering. FASE 2017. Lecture Notes in Computer Science, vol 10202. Springer, Berlin, Heidelberg. https://doi-org.www.bibproxy.du.se/10.1007/978-3-662-54494-5_6

Love, C. (2018). Progressive web application development by example: Develop fast, reliable, and engaging user experiences for the web. ProQuest Ebook Central https://ebookcentral.proquest.com/

Sandberg, E. (2021). Evaluating Blazor : A comparative examination of a web framework (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-173239

Skansholm, J. (2008). *Skarp programmering med C#*. (1. uppl.) Lund: Studentlitteratur.

Szelogowski, D (2021). Optimize data cube visualization for web applications. Performance and user-friendly data aggregation. Retrieved from https://arxiv.org/abs/2101.00171

Wang, Y., Cheng, K. S., Song, M., & Tilevich, E. (2019). A declarative enhancement of JavaScript programs by leveraging the Java metadata infrastructure. Science of Computer Programming, vol 181, pages 27-46. https://doi.org/10.1016/j.scico.2019.05.005.

Web Pages

Angular.io (2021, April). Angular versioning and releases. Available at: https://angular.io/guide/what-is-angular (Accessed 23 April. 2021)

Angular.io (2021, April). What is Angular?. Available at: https://angular.io/guide/releases (Accessed 13 April. 2021)

Atwood, J. (21 May 2007). JavaScript: The Lingua Franca of the web. *Coding Horror*. https://blog.codinghorror.com/javascript-the-lingua-franca-of-the-web/

Guru99 (2021, April). Typescript vs Javascript: What's the difference? https://www.guru99.com/typescript-vs-javascript.html (Accessed 14 April. 2021)

Lighthouse (2021, April). Tools for Web Developers. https://developers.google.com/web/tools/lighthouse (Accessed 14 April. 2021)

Microsoft.com, (2021, April). Blazor. Available at: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor (Accessed 14 April. 2021)

Tiobe.com, 2021. TIOBE Index for May 2021. Available at: https://www.tiobe.com/tiobe-index/ (Accessed 10 May. 2021)

Web.dev (2021, May). Cumulative Layout Shift. Available at: https://web.dev/cls/ (Accessed 7 May. 2021)

Web.dev (2021, May). First Contentful Paint. Available at: https://web.dev/first-contentful-paint/ (Accessed 7 May. 2021)

Web.dev (2021, May). Largest Contentful Paint. Available at: https://web.dev/optimize-lcp/ (Accessed 7 May. 2021)

Web.dev (2021, May). Speed Index. Available at: https://web.dev/speed-index/ (Accessed 7 May. 2021)

Web.dev (2021, May). Time to Interactive. Available at: https://web.dev/interactive/ (Accessed 7 May. 2021)

Web.dev (2021, May). Total Blocking Time. Available at: https://web.dev/tbt/ (Accessed 7 May. 2021)