

## Web application for user acquisition through Search Engine Optimization



BACHELOR'S THESIS 2018

# Web application for user acquisition through Search Engine Optimization

Melina Andersson  
Kristoffer Ek  
Adrian Lindberg  
Long Nguyen  
Alex Sundbäck  
Jonathan Sundkvist  
David Söderberg



## Web application for user acquisition through Search Engine Optimization

- © Melina Andersson, 2018
- © Kristoffer Ek, 2018
- © Adrian Lindberg, 2018
- © Long Nguyen, 2018
- © Alex Sundbäck, 2018
- © Jonathan Sundkvist, 2018
- © David Söderberg, 2018

Supervisor: Olof Torgersson

Associate professor, Interaction Design division, Department of Computer Science and Engineering.

Examiner: Morten Fjeld

Professor, Interaction Design division, Department of Computer Science and Engineering.

Bachelor's thesis DATX02-18-18

Department of Computer Science and Engineering

Chalmers University of Technology

SE-412 96 Gothenburg

Telephone +46 31 772 1000

Gothenburg, Sweden 2018

## Abstract

The project this theses is based upon aims to develop a web application in order to study how one can utilize it to acquire new users to a mobile application. This problem can further be divided into two main areas of investigation. First of all, how to maximize the incoming traffic to the web application by optimizing the visibility online. Secondly, how to convert visitors into users of the mobile application. In order to evaluate this in practice, the development of the web application was done in close collaboration with Forza Football, a company that provides world wide live coverage and statistics of football leagues with over 2 million monthly mobile application users. Techniques in the fields of User Acquisition and Search Engine Optimization were implemented to examine the effectiveness of the strategies researched. Tracking the difference in incoming traffic and user interactions showed that one of the most important additions to the application was the introduction of keywords and meta descriptions. Factors such as page loading speed, quality content and engaging design proved to be successful for both page views and bounce rate. Even though the deployed application was in an early stage when the results were measured, the visitors that did visit were from all around the world and had a sport interest which indicates that the right type of users were attracted. The results showed that a web application that acquires users to a mobile application is a promising strategy for a company to grow.

Keywords: user acquisition, SEO, web application, tracking, football, live data, UX

## Acknowledgements

We want to thank Forza Football for giving us this opportunity and a special thanks to Andreas Rolén, Head of Growth at Forza Football, for his contribution.

## List of Abbreviations

**SEO** Search Engine Optimization  
**JSON** JavaScript Object Notation  
**HTML** HyperText Markup Language  
**AWS** Amazon Web Services  
**CSV** Comma Separated Values  
**SQL** Structured Query Language  
**DOM** Document Object Model  
**JSX** JavaScript XML  
**XML** EXtensible Markup Language  
**IDE** Integrated Development Environment  
**SSL** Secure Sockets Layer  
**URL** Uniform Resource Locator  
**UX** User eXperience design  
**REST** REpresentational State Transfer

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Purpose . . . . .	2
1.3	Problem Statement . . . . .	2
1.3.1	Incoming traffic . . . . .	2
1.3.2	Converting visitors . . . . .	2
1.3.3	Handling the data . . . . .	2
1.3.4	Frameworks, Libraries & Techniques . . . . .	3
1.4	Goals . . . . .	3
1.5	Delimitations . . . . .	3
1.6	Ethical Aspects . . . . .	4
<b>2</b>	<b>Theory</b>	<b>5</b>
2.1	User acquisition . . . . .	5
2.2	User acquisition online . . . . .	5
2.3	Online strategies for gaining customers . . . . .	6
2.4	Search Engine Optimization fundamentals . . . . .	6
2.4.1	Relevant content . . . . .	7
2.4.2	Outbound and inbound links . . . . .	7
2.4.3	Meta descriptions and title tags . . . . .	7
2.4.4	Fast page loading speed . . . . .	7
2.4.5	Security . . . . .	8
2.4.6	Sitemap . . . . .	8
2.4.7	Responsiveness . . . . .	8
2.5	User Experience . . . . .	8
2.5.1	Layout . . . . .	9
2.5.2	Mobile Layout . . . . .	9
2.5.3	Chunking . . . . .	9
2.5.4	Mental models . . . . .	10
2.5.5	Entry points . . . . .	10
2.5.6	Response times . . . . .	10
<b>3</b>	<b>Method</b>	<b>11</b>
3.1	Workflow . . . . .	11
<b>4</b>	<b>Libraries and Tools</b>	<b>12</b>



4.1	Libraries . . . . .	12
4.1.1	React . . . . .	12
4.1.2	Next.js . . . . .	12
4.1.3	Node.js . . . . .	12
4.2	Tools . . . . .	13
4.2.1	Google Analytics . . . . .	13
4.2.2	MySQL . . . . .	13
4.2.3	GraphQL . . . . .	13
4.2.4	Docker . . . . .	14
4.2.5	Amazon Web Services . . . . .	14
4.2.6	Service Worker . . . . .	15
<b>5</b>	<b>Implementation</b>	<b>16</b>
5.1	First Iteration . . . . .	16
5.1.1	Setting up the database . . . . .	16
5.1.2	Design 1.0 . . . . .	16
5.1.3	Snackbar . . . . .	18
5.1.4	Optimizing SEO . . . . .	19
5.1.5	Optimizing performance . . . . .	19
5.1.6	Deploying . . . . .	19
5.2	Second Iteration . . . . .	20
5.2.1	Design 2.0 . . . . .	20
5.2.2	Fetching live data . . . . .	21
5.2.3	Optimizing the GraphQL-wrapper by caching . . . . .	22
<b>6</b>	<b>Result</b>	<b>24</b>
6.1	User Interface . . . . .	24
6.1.1	Sidebar . . . . .	24
6.1.2	Match view . . . . .	25
6.1.3	Team view . . . . .	27
6.2	Visibility and Incoming Traffic . . . . .	28
6.2.1	Did it rank highly for core themes and target phrases? . . . . .	28
6.2.2	How much traffic was driven to the site? . . . . .	30
6.2.3	Did it attract the target audience? . . . . .	31
6.3	Content and User Interactions . . . . .	32
6.3.1	Did the visitors bounce or did they find the content engaging and interacted with it? . . . . .	32
6.3.2	Did the content fuel goal completions/conversions or drive sales? . . . . .	33
6.3.3	Did the web application perform well enough? . . . . .	33
<b>7</b>	<b>Discussion</b>	<b>35</b>
7.1	User Interface . . . . .	35
7.2	Visibility . . . . .	35
7.3	Handling the Data . . . . .	36
<b>8</b>	<b>Conclusion</b>	<b>38</b>

<b>Bibliography</b>	<b>39</b>
<b>A Appendix 1</b>	<b>I</b>
A.1 Feature Branch . . . . .	I
A.2 MySQL . . . . .	I
A.3 GraphQL . . . . .	II
A.4 Docker . . . . .	II
A.5 Figures . . . . .	III

# 1

## Introduction

Nowadays, football fans have a wide variety of mobile applications to choose from in order to keep track of results, upcoming matches and statistics. When not being able to follow games from the stands or on TV they instead rely on mobile devices to keep track of real-time events and statistics. These applications are in general free of charge but expose the user to advertisements from which the companies generate income. Therefore, the companies rely on not only a stable user base but also on the acquisition of new users in order to increase their revenue. One of the methods used to gain new users is to provide a web application with some of the functionality found in the mobile application. The companies can then increase their revenue by the use of advertisements on the web application or by encouraging the user to download the application.

### 1.1 Background

Göteborg based company Forza Football provides one of many options when it comes to mobile applications providing football related statistics. They started out in 2012 with a web-based product but quickly shifted to only having mobile applications for iOS and Android. The reason was the tougher competition on the web and a faster-growing user base on mobile. Today, they have over ten mobile applications with the application called Forza being the most popular one providing live coverage and statistics for more than 800 different tournaments around the world[1].

Despite already having a large number of users they, like most companies, strive for growth. One of the methods for acquiring users is the use of web applications. With over four billion active internet users worldwide[2], a company with no digital presence is close to non-existent. Forza Football's current website only serves a branding and marketing purpose. In other words, none of their data concerning football is available on the web which implies that valuable internet traffic and searches regarding football are missed out on.

Forza Football has realized that one way to grow their user base is by extending their web presence with a web application containing some of the functionality found in their mobile application. Thus, when a user searches on a search engine for a key-phrase like "Arsenal vs. Manchester United", the web application in question should appear as an alternative among the result. Intrigued by the functionality and experience, the ultimate goal is that the user downloads and starts using the

mobile application.

This thesis will cover the development of such a web application. The development was done in close collaboration with Forza Football who provided the data necessary and technical advice when required.

## **1.2 Purpose**

The purpose of this thesis is to investigate how a web application can be utilized to acquire new users to a mobile application. In order to achieve this, the implementation of the web application will focus on two different areas. First of all, having a widespread visibility online so that new users will find the web application. Secondly, providing content and a user experience that encourages users to download the mobile application.

## **1.3 Problem Statement**

The main problem is how to construct a web application to acquire new users to a mobile application in the most efficient way. This problem can be derived into several sub-problems.

### **1.3.1 Incoming traffic**

A high amount of incoming traffic is beneficial for a web application of this project's purpose. Thus, questions like "how does one rank high in a search engine" and "which keywords/key-phrases does one want to focus on" need to be further investigated. The basic functionality of search engines and especially how they index pages have to be taken into consideration throughout the whole development process.

### **1.3.2 Converting visitors**

Converting visitors to become users of the mobile application is crucial in order to fulfill the purpose of the project. How does one construct a web application to make the user want to download the mobile application? Further investigation has to be put into what key factors are important in order to convert the users.

### **1.3.3 Handling the data**

The data provided from Forza Football contains information about the four greatest football leagues in Europe, namely Premier League (England), Bundesliga (Germany), Serie A (Italy) and La Liga (Spain). All information about the matches, players and teams are covered in the data. How does this data gets converted to a fully working web application?

### 1.3.4 Frameworks, Libraries & Techniques

Today, there are thousands of frameworks and libraries that can be used in the context of a web application. Many of them solve the same problem but with different approaches. Which approaches are best suited for this particular case?

Another problem with the implementation of the web application is how to approach the difficulties with real-time data. Are there any libraries which can be helpful? How will the page update itself when it receives new data? What is considered best practice when dealing with real-time systems?

## 1.4 Goals

Together with our supervisor at Forza Football, six goals were stated at the start of the project. These goals were based on research and experience from our supervisor, whose main area is company growth.

- Web application appears on at least the 2nd page when searching for a random match (within the scope of the project) on Google
- Web application appears on at least the 3rd page when searching for a random team (within the scope of the project) on Google
- Web application appears on at least the 3rd page when searching for a random player (within the scope of the project) on Google
- Web application appears on at least the 4th page when searching for a random tournament (within the scope of the project) on Google
- 3% of the visitors of the page go to the App Store page for the Forza application.
- 25% that goes to App Store downloads the Forza application.

## 1.5 Delimitations

The web application will focus on matches, teams, players and tournaments. The search queries considered relevant to the problem statement, and ultimately the result, are restrained to these areas only, and not for queries concerning e.g. football related news. Further delimitations have been made when it comes to the web application's rank on said target queries. Depending on the search query the goal has been to appear on one of the four first result pages. This is motivated by the fact that there are plenty of well-established companies and media houses that already provide the same services as this particular web application. Therefore, it's unreasonable to expect to outrank these within the time-frame of this project. Due to Google's market dominance, the web application will be developed with their guidelines in mind. Furthermore, all queries made for the sake of analyzing the web application's progress and result concerning SEO will be using their search engine.

## 1.6 Ethical Aspects

In order to gather data on for example how many visitors arrive to the web app or on how many stay, tracking analytics is used. The collected tracking data is anonymized and the analytics service does not save IP-addresses (Internet Protocol address) or MAC-addresses (Media Access Control address) so there is no direct connection with data to an individual.

# 2

## Theory

The result of the project is dependant on choices made in terms of how to approach User Acquisition as well as Search Engine Optimization. Therefore, studies is required in these areas. This chapter will cover the theory found relevant to the purpose of the project.

### 2.1 User acquisition

Every business requires customers in order to survive, this involves acquiring new customers as well as ensuring the return of current customers. This is often referred to as “customer acquisition”, or in this case user acquisition, since the customers of the web application are not paying for a product or service directly[3]. Instead, their usage of the application, in this case the downloading and usage of the Forza mobile application through the web application, generates income from advertisements. What makes customer acquisition relevant is that it is one of the most cost-effective ways to increase growth rapidly[3].

### 2.2 User acquisition online

In order for a business to be found by customers, marketing is often required in some way. Marketing is commonly divided into two categories: inbound and outbound. Outbound marketing consists of trying to gain new customers by paying for advertisements on different channels such as TV, newspapers or websites in order to “push” a product or service onto people. Inbound marketing, on the other hand, is focusing on the quality of the content to be satisfying enough to acquire the customers on its own[4]. If this can be done, the customer is more likely to remain a customer to a lower cost than traditional marketing such as advertisements. Therefore, the focus in this project will be on this form of marketing.

Focusing on inbound marketing, the main question is how the customers are supposed to find quality content when they are not aware of its existence. Today, search engines are used around the world in order to find content online for specific keywords. Around 3.5 billion searches are made every day on Google, the most used search engine online[5]. The problem for today’s businesses is that there might be billions of results for every search, making it difficult to be found by the customers.

This is where SEO comes into play. The search engines organize the results according to the content on websites in order to display the most relevant pages first. SEO are techniques for optimizing the content on a website to reach the top of that list of results, and by that acquire visits from the users searching[6].

## 2.3 Online strategies for gaining customers

SEO can be utilized to gain new customers, but how should the optimization be done on a website in order for it to be considered relevant by the search engines? How do search engines work and what do their algorithms take into consideration when indexing millions of sites? These are not easy questions since the answer varies from search engine to search engine, and also changes over time. Another thing to consider when optimizing a site for search engines is that the terms searched for by the users are unknown. It is possible to look at statistics over common search terms but they will be historical, users will not necessarily search in the same patterns in the future. Knowing that these search terms are changing continuously and how crucial they are, SEO becomes one key to success in gaining new customers. However, it does not ensure any loyalty of the customers visiting the website. After all, the goal of this project is to acquire more users to Forza's mobile application. Therefore, the quality of the content is important, and a tool to find out if the content on a website is good or not is the use of tracking. Tracking, in this case, means collecting data on the user's actions while using the page, but also before entering and leaving it[7]. The tracking data might contain anything from what the keywords searched for was, what nationality the user had, how the user navigated the application as well as how long the user stayed on the page. Gathering information like this helps to improve the content and layout, due to the possibility to see the result from various theories and techniques. As a result, the improved content helps to create satisfied and thereby loyal customers.

## 2.4 Search Engine Optimization fundamentals

As mentioned earlier, the best practices regarding SEO are not consistent. Since different algorithms are used by different search engines and they often change, the implementation of SEO in this project is focused on the guidelines from Google the dominating search engine with almost 92% of the world's market share[8]. Due to this Google has a great impact on the practices of SEO and should therefore be taken into consideration.

The most general guidelines[9] that have been taken into account in this project are:

- Relevant content
- Relevant keywords in URL, titles and text
- Outbound and inbound links
- Meta tags and meta descriptions
- Fast page loading speed



- Responsiveness

### 2.4.1 Relevant content

What defines relevant content and keywords in URL, titles and text? Instead of asking every user whether or not they found the content relevant, measuring the time a visitor spent on a web page gives an indication. If users stay on a web page longer this indicates that they find the content engaging and valuable which in turn indicates to the search engine that the page should be ranked higher. Since search engines mainly generate profit by selling advertisements and promoting results[10], the more traffic they receive, the more money they make. Hence, they strive to optimize the search results for users in order for them to continue using their service. Analyzing the time spent on a web page is one way to measure how relevant a web page is, another is to measure bounce rates. The bounce rate is the percentage of single-page sessions in which there was no interaction with the page[11].

### 2.4.2 Outbound and inbound links

Outbound, inbound and internal links are also part of the ranking system. This can be thought of as a graph, where each node is a website and each edge is a link to that website from another website. A link from a website to another is called an inbound link. Each edge has a value representing its importance, for example, a link from a highly visited site like *www.bbc.com* is valued more than a link from a less visited site like a blog or a company's homepage. Many links from quality sites will increase the ranking of the site being linked to[12]. Furthermore, links from a website to other websites, so called outbound links, is profitable for SEO. A search engine will interpret a website as a source of information and will rank it higher than websites that lack outbound links.

### 2.4.3 Meta descriptions and title tags

When a search engine crawls (analyzes the content of a web page) it uses its meta description and title tags in order to decide if the content is of relevance or not for the user[13]. The former describe and summarize the data on a specific page as accurately as possible with the aid of keywords and relevant phrases[14]. If the content on separate pages varies on a website, the meta descriptions should be dynamic and vary accordingly. Title tags, one of the most important components of good SEO are also meta descriptions that won't appear on the actual page[15]. However, they can be seen at the result page of a Google search and at the top of the browser if the page is entered.

### 2.4.4 Fast page loading speed

In 2010 Google announced that they take page loading speed into account when ranking websites, making a slow website being valued lower than a faster one[16]. According to Google's internal studies, a visitor spends less time on a website if it is

responding slowly, so by promoting websites that respond faster search engines will increase the satisfaction of their users and in turn also their own revenue.

### 2.4.5 Security

SEO can be boosted by making sure the web page is safe and encrypted. This reduces the chance that a leak would occur, and if it did, no registered users would be affected. This is done by SSL (Secure Sockets Layer) which encrypts all traffic from and to a website. In 2014 Google posted on their online security blog that they started taking SSL usage in their ranking algorithm which will actively worsen non-SSL websites rankings[17].

### 2.4.6 Sitemap

A sitemap is a file, usually written in XML-format, which specifies the structure of a web application. This makes sure that search engines do not forget to crawl any of its pages, which is helpful when there is not a lot of external links referring to the site[18].

### 2.4.7 Responsiveness

Google themselves has, in their developer pages, explained that they prefer responsive design rather than web pages designed for mobile with another URL[19]. Non-responsive web pages usually have a URL similar to *m.example.com*, which is a mobile designed version of the web page. The benefits of responsive design, according to Google, is that it helps their algorithms with indexing properties of the page and saves time when the crawler does not have to visit a mobile version as well[19]. A responsive design combined with a fast page loading speed provides an improved user experience.

## 2.5 User Experience

Even though there is no exact definition of User Experience (UX), the consensus is that it is the process of designing a product with the user in mind with the ultimate goal to provide a pleasant experience[20][21]. This experience should be in line with the user's expectations of the web application when it comes to both content and behavior. In this case, it is reasonable to assume that a visitor is looking for live updates or statistics on a match, scheduled for a specific team, info about a tournament or similar. With this in mind, the challenge is how to present this information in an intuitive way.

A term often used when talking about the user experience is *the cognitive load* which refers to the amount of brain power a user have to utilize in order to process the presented information[22]. The cognitive load imposed on the user should be minimized and there are plenty of techniques one can use to achieve this. With the nature of the proposed web application in mind, especially its vast amount of

data, there were certain aspects of the UX that would be of special importance: organization of the data and simple navigation between entities.

### 2.5.1 Layout

With the events, positions and statistics being the most important components of an entity within the web application it was desired to put most of the visual emphasis on these elements when designing the layout. But when lacking data on what content that would be of most interest to the users it is important not to create an imbalance in the way the content is presented and highlighted. Two common design patterns that can be utilized in such situations are *Center Stage* and *Grid of Equals*. Center Stage is used when some content is central to the interface. It is implemented by dedicating a majority of the available space to its most significant content and placed in such a way that it cannot be missed. Grid of Equals, on the contrary, is implemented by organizing content in a grid layout and giving every item the same visual weight, indicating that they have the same importance[23]. Combining these by putting a Grid of Equals within a Center Stage communicates to the user that the centred items are important but at the same time of equal importance relative to each other. However, Centre Stage and a grid with multiple columns is only fitting when the browser or screens size allows it. When visiting the web application with a smaller browser size or on a mobile device the layout would have to adjust dynamically to the screen size (see section 2.5.6).

### 2.5.2 Mobile Layout

If a user visits the web application on a mobile device, or just wish to scale down their web browser, the web application should have to adjust accordingly. With limited screen size, the design will have to be presented with remembrance. A common design pattern to use when dealing with narrow screen widths is *Vertical Stack* where content, otherwise placed side by side, is stacked vertically[23]. The content should be placed in a logical order so that the most interesting items are placed at the beginning of the stack. If a sidebar is included in the design it should be hidden until actively called into view by the user in order to keep focus on the main content. However, it is important to somehow indicate that the sidebar actually is available to the user[24].

### 2.5.3 Chunking

Chunking is the concept of splitting up the content of the graphical interface into smaller pieces. This makes it easier for the users to process the content since they will not have to memorize all of it at the same time. By chunking and grouping specific content in a logical way the user can focus on the most interesting parts momentarily[25]. In this web application, this could mean that statistics and line-ups for a match are grouped separately within a match page since they are not directly connected but both still important aspects of a match.

### 2.5.4 Mental models

One of the most important aspects to consider when designing an interface is the users' preexisting mental models. Mental models can be described as how the users believe and/or expect a web page to behave when interacting with it. These mental models are formed from previous experiences with other websites[26]. For example, web shops often use the design pattern *Feature, Search, Browse* which includes a sidebar, a header with a search field and a centred main view with the product[23]. Since many web shops use this layout, deviating from it is likely to go against the user's mental model and therefore increase the cognitive load.

### 2.5.5 Entry points

For someone unfamiliar with the web application it is desirable to make the layout as intuitive and easy to use as possible. This is of special importance when presenting the user with a lot of information and options[23]. In these cases, it should be obvious to the user what every choice will result in and how to reach their objective. A visitor to this particular web application is likely to be interested in a specific match, but might also want to get updates on a related one. Therefore it's important to design the layout in a way that follows the users mental model as accurately as possible to minimize cognitive load when navigating to another match or entity entirely.

### 2.5.6 Response times

With broadband and fast internet connection being common, internet users are used to short load time and seamless navigation. Thus, having a web application that forces the user to wait longer than even just a few seconds for its content to load is likely to cause disruption and frustration[27]. In order to keep the waiting short, it is important to not overload the interface with for example unnecessary animations or large image files.

# 3

## Method

### 3.1 Workflow

During the entire thesis, the team worked agile in weekly sprints with every week starting with a meeting at Forza Football to discuss the progress. Since the purpose of the project was to acquire new users to the mobile application, an important step in the process was to retrieve data early on for benchmarking purpose. Therefore, an initial decision that the development of the web application should be divided into two larger iterations was made. The first iteration would present a simple initial sketch of the web application given the first data provided by Forza Football. Since the data did not include any live score or recent information, the focus was instead put on building a good infrastructure and start tracking, as well as measuring the website traffic. With that in place, the second iteration would shift focus to developing and implementing a new design with better and richer content including live result and current news. This would be built on top of the infrastructure from the first iteration.

As mentioned above, the team carrying out this project has worked according to an agile working method called Scrum. Agile workflow is about dividing the project into multiple smaller steps so there can be a continuous delivery from the start instead of delivering everything in the end. Moreover, Scrum is an approach where the team works in sprints that are set to a given length that suits the projects time plan, in the case of this project the sprints is set to one week. Each sprint, in this case each week, starts with a meeting where the team together with a Scrum leader plans the following week by prioritizing which tasks that needs and can be implemented[28]. Additionally, to facilitate the development process, a workflow called *Feature Branch* has been used continuously throughout the project. This workflow is further explained in Appendix A.2.

# 4

## Libraries and Tools

This chapter describes how libraries and tools were used to benefit the development of the web application. Using libraries saves time when one does not need to reinvent the wheel. The tools helped with organizing data and creating a uniform environment for the developers.

### 4.1 Libraries

#### 4.1.1 React

One of the frameworks used were React that is developed by Facebook Open Source[29]. React compiles JavaScript code into HTML on the client. The usage of React in this project was partly because of its capabilities for scalable web applications, but also because it is one of the fastest web frameworks with its use of Virtual DOM. DOM, which stands for Document Object Model, is a programming interface that represents the structure of a web page as nodes and objects[30]. Virtual DOM is a copy of the DOM in a browser. Basically, when a change is made React checks the difference between the Virtual and the real DOM and only updates the elements in the latter that have been updated in Virtual DOM. This makes React very fast compared to other frameworks that update the whole page every time a change is made. As mentioned earlier, fast page loading speed is one of the factors that improve user satisfaction which made React suitable for this project.

#### 4.1.2 Next.js

Another technique that decreases the page loading time is server-side rendering. Server-side rendering is when the code gets executed on the server and sends the result back to the client. Normally, JavaScript code is compiled into HTML files on the client, with client-side rendering, but server-side rendering compiles JavaScript into HTML on the server. In order to implement server-side rendering a JavaScript library called Next.js were used. Next.js is a framework developed by Zeit made for server-rendering or statically-exporting React Applications[31].

#### 4.1.3 Node.js

Node.js is an open-source and cross-platform environment that executes JavaScript server-side and is mainly used to create web applications[32]. Node.js is a great platform for basing a JavaScript web application on because of it's vast supply of

libraries that can be used. Together with Express, a web application framework for Node.js, a reliable back-end API can be built which also provides a feature that automatically transforms SQL-files from MySQL database into JSON-object for the web application.

## 4.2 Tools

### 4.2.1 Google Analytics

Some tools were used for measuring how successful the web application was at its tasks. A tool for measuring user tracking was Google Analytics[33]. Google Analytics is capable to track actions on the page as well as measuring where the visitor came from before visiting the page. This project used tracking for measuring how many visitors that went to download Forza application. This was done with a simple event tracker when clicking on a link to App Store/Google Play or for Desktop users; Forza's homepage. An event tracker fires an event when the visitor, for example, presses a link. The request is then sent to a category, label and a value (optional). With that data, it becomes easier to see where visitors click and to find faults in the design. The data makes it possible to map out deviations and check where the average visitor clicks first and so forth.

### 4.2.2 MySQL

Since a stable and efficient database is preferable for any web application, several alternatives were discussed in an early stage of the project. At the beginning of the development the data received from Forza was a SQL dump. A SQL dump contains all the data from the whole database in form of a list of SQL statements. The database system chosen was finally MySQL since it is one of the most used relational database systems, both at Forza and globally. For more detailed information about MySQL, see Appendix A.2.

The main reason why the database has a critical role in both the development of the web application and SEO, is because server-side rendering depends a lot on it. A stable and efficient database ensures the possibility for the search engine to index and crawl the pages, which grants the pages a higher ranking result. By containing all necessary data, the database helps to solve the problem of generating new pages as well as erasing and adjusting old pages when new data is provided by Forza Football. Later on, having access to Forza Football's API together with the database eliminates the demand of waiting for new data before performing any adjustment. This establishes an automatic process of fetching new slow-data and adjusting after changes of the web application.

### 4.2.3 GraphQL

The query language GraphQL was used in this project. More information about GraphQL may be found in Appendix A.3. The differences between the data provided

with MySQL (slow data) and the data provided with GraphQL (live data) are mainly that the slow data is used to build the structure of the application, by building routes that are available for the user to visit. Therefore, this data should contain the necessary data to make an initial render of the web application, and provide the data needed that's used by the meta tags. For example, if a match-page should render some slow data, it needs basic relevant information about the match, such as the ID of a match, when the match is bound to start and which teams that are playing. The slow data doesn't need to update often since the basic data about the matches doesn't change frequently. The live data, however, includes all content about the current state of the match, for example the match score, live news and statistics. In the GraphQL-wrapper that's used in this project, it's also possible for the client to subscribe for new live events, which means that the client automatically gets updated when the GraphQL-wrapper receives new data.

#### 4.2.4 Docker

For many reasons, maintaining a similar development environment might be troublesome and expensive, especially when different types of operative system and hard wares are in use. The reason is that the above-named differences might cause multiple problems. Therefore, Docker is in use in order to construct a similar and smooth development environment for all the developers. How Docker works is explained in Appendix A.4.

#### 4.2.5 Amazon Web Services

Amazon Web Services (AWS) is a part of the online shopping company Amazon that provides on-demand cloud computing platforms[35]. With over one million customers, including well-known companies like Spotify, Airbnb and Yelp[36], AWS contains more than 90 services that should include everything needed to build a desired IT infrastructure. These services are implemented in server farms placed around the world that is maintained by Amazon. Amazon is then charging their customers based on usage and what kind of service that's being used. Data from the first quarter of 2018 shows that AWS owns 33% of all cloud hosting in the world[37].

One of the services that were used during the project is called Amazon S3. S3 is an object storage where it's possible to store any amount of any data. This is one of the most popular services that AWS are providing[38] since it's a service that suits a wide range of customers, no matter if there are individuals who only want to store a small website to companies that want to store a large amount of data. During the project, S3 was mainly used to store SQL-dumps that the web application lately used for its rebuilds.

To host the project, both the web application and the GraphQL-server, a service called AWS Elastic Beanstalk was used. AWS Elastic Beanstalk handles automatically how many cloud computing instances that are needed for the application to run smoothly for every visitor, which is also known as auto-scaling. The cloud computer



instances for this project is called AWS EC2.

An important step of the deployment process was to find a simple way of transferring the Docker containers that were created on the team's local machines, to AWS and the EC2 instances. To achieve this bridge, AWS Elastic Container Registry (ECR) was used. ECR is a service that makes it possible for the local machines to push their Docker containers to a container registry. ECR then notices that a new version of the application was pushed, and tries to deploy that version to the AWS Elastic Beanstalk (described above). If the team happened to push a broken Docker container that didn't work out as expected, ECR made it really simple to roll back to an older version of the application that the team knew was working.

Lastly, a service called Amazon CloudFront is used as a content delivery network (CDN). A CDN is a network of proxy servers placed in different places around the world, which delivers the content to the users as fast as possible by using the local cache of static content. In the project, all images are cached for 30 days which makes the delivery of these images faster. CloudFront also makes sure that the custom domain of the application, `livescore.forzafootball.com`, is pointing correctly to the Elastic Beanstalk application.

#### **4.2.6 Service Worker**

In order to deliver content to the user, regardless of the user network state, a script called Service Worker was used. A Service Worker is a proxy that is placed between the web application and the user's network, making it possible to control network requests and respond to them in different ways[39]. Therefore, a service worker is able to provide the user with a cached version of the web application if the user's network connection is unavailable. This delivery of cached content is made without the user's device being connected to the server that is hosting the application, which saves a great number of network requests to the server and also increases the user's performance.

# 5

## Implementation

This chapter is a somewhat chronological description of the process of implementing the web application. It will not include technical descriptions of any libraries, frameworks or tools that have been used but rather refer to appropriate section in the *Libraries and Tools* chapter whenever needed.

### 5.1 First Iteration

The following sub-sections describes implementation steps in the first iteration.

#### 5.1.1 Setting up the database

The focus during the first iteration was mostly on building a steady project structure as well as putting up a database and adjust it to the format of Forza's data. During the project, this was done using a MySQL database and the data were provided by Forza in *.sql*-format, which made it simple to import the data. With the database up and running, the next step was to provide an easy way to access it when building the actual web application. Based on the data in the database it was now possible to define all available URL-routes. A simple REST-API[40] was created by specifying some of these routes that presented the data from the database in JSON-format, which would be the format used when building the front-end of the web application. The REST-API was defined using Node.js (see section 4 *Libraries*). The use of a REST-API also prevent SQL-queries from being sent directly to the database which could possibly be a security issue since hackers then could get a dump of the database content by performing a SQL injection.

#### 5.1.2 Design 1.0

The provided data during the first iteration was limited to a small selection of matches from a single league, and because of this, designing the page was a challenge. A major part of the pages was empty, without any data to show (see figure 5.1). The design was based on the research on UX but inspiration was also found in the mobile application (see figure 5.2). This was because at first the web application was merely considered an extension of the application. The main components of the first design were:

- A sidebar containing relevant matches to choose from.
- A main container showing the content of the selected match, tournament or player. This component could be divided into three sub-components:
  - A 'jumbotron' at the top of the page, showing the most relevant information for the current page.
  - A navbar to enable logical grouping of different information
  - A container showing information depending on the selected tab in the navbar.

The choice of having a sidebar displaying matches originated from the design of the mobile application where the list of matches was considered the 'homepage' and the main mean of navigation between matches. Due to the possibilities of the larger format of a web browser compared to a mobile device, it was decided that the sidebar should be statically placed on the left side of the page as long as the screen width was sufficient. This choice was further founded in the theory concerning mental models and entry points since sidebars are a common mean of global navigation and by always making it accessible the user wouldn't get lost. If the web browser would be scaled down or viewed on a mobile device, the sidebar would be hidden until the user actively chose to show it again.

The disposition of the main component in relation to the sidebar followed the Center Stage design pattern. 80% of the screen width was assigned to the former making it the dominant section of the interface. Its sub-components followed the arrangement of the mobile application and the idea was that the layout would remain the same even when the content changed in order to minimize cognitive load for the user.

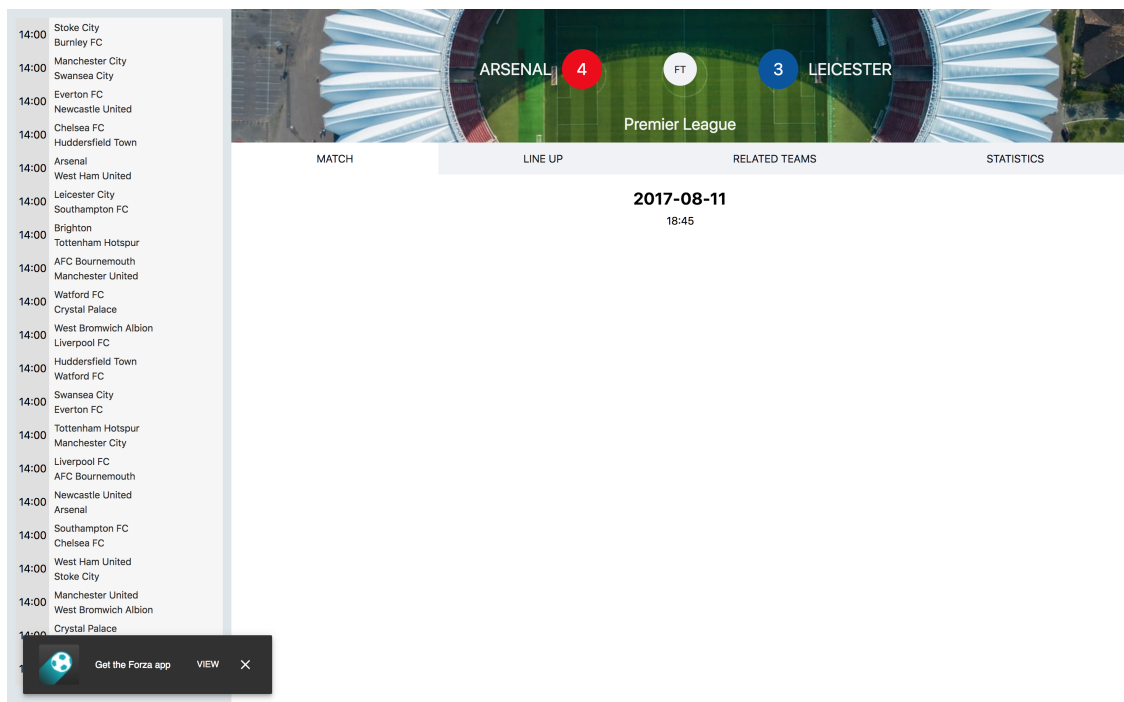
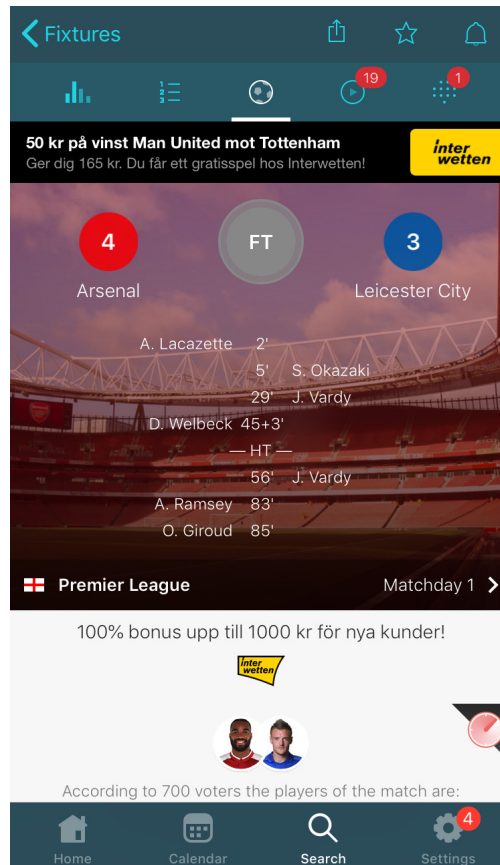


Figure 5.1: The first iteration of the web application



**Figure 5.2:** Forza Football’s mobile application

The next step was to implement the design and build the actual web application. It was decided to use the JavaScript framework React (see section 4 *Libraries*). Moreover, one of the goals stated at the beginning of the project was that the web applications should rank high on search engines as early as possible in order to be able to measure and compare results at the end of the project. In order to achieve this, the web application would have to be built with SEO in mind from the start. This objective became a challenge since all search engines were not able to crawl data from a web application built with JavaScript based frameworks such as React. To solve this problem it was decided to use Next.js (see section 4 *Libraries*), when approaching this challenge. This made it possible to server-side render the data straight away using one of the framework’s function, which executes on the server and returns a populated page to the client.

### 5.1.3 Snackbar

The most crucial component of the web application was the so called *snackbar*, seen at the bottom left of figure 5.1. The snackbar should appear for the user with a given interval and present the suggestion of downloading the mobile application. When clicked on in desktop mode, the user would be redirected to Forza Football’s website, specifically the sub-page highlighting the content of the mobile application.

On mobile devices, the snackbar would appear at the top of the screen and redirect the user directly to App Store/Google Play or if the user already has the application installed, to the corresponding page within the application. However, it was important not to spam the user with the snackbar every single session since that could prove counterproductive due to irritation and thus negatively impact UX and consequently the SEO. To avoid such scenario the snackbar is only shown once every session start.

#### 5.1.4 Optimizing SEO

In parallel to implementing the front-end of the web application, crucial changes were made to the inner logic of it. For the sake of SEO, methods for generating dynamic meta descriptions, title tags and URL:s were implemented. For example, for a match page, the meta description would be customized to include the teams involved, the tournament they play in, the date and phrases stating that the content also includes stats and news among other content. The URL for every page of the web application was also customized and given appropriate formatting so it would be as descriptive and accurate as possible based on its content. Furthermore, using the given data in the database, a sitemap for every single page that was available for the user could be created. Since the web application would be brand new this was crucial since there would not be any external links to it initially and therefore search engines might overlook some of the available content when crawling the page.

#### 5.1.5 Optimizing performance

By the end of the iteration, it was decided that the web application would use the system font instead of a custom one. The motivation for this choice was to improve performance and response times since the site no longer would require downloading a custom font for its pages.[41] This, in turn, would lead to better SEO since site performance is another factor search engines take into account when indexing web pages. Additionally, better response time improves the UX which long-term also boosts SEO.

#### 5.1.6 Deploying

A crucial part of the project was to set up a simple way of transferring the project from the team's local machines to the servers that hosted the project. Since all of the team members used Docker containers to develop, everyone in the team worked with the same environment which gave confidence that it should work with the same result for everyone. The best way to deploy this project would, therefore, be to transfer the Docker containers to the hosting cloud computers to ensure the desired result after deploying. However, the Docker containers used during development was unnecessarily big in memory size, since they contained a lot of debugging and development-specific tools that weren't needed in production.

To solve this issue, two Dockerfiles were written. Both of them were based on Node 9.5 Alpine[42], but in the production-ready Dockerfile only the actually necessary

packages needed for the container was installed. In the development container, the desired feature would, for instance, be to auto reload the application when any file in the project was changed. However, this was not a feature that would be needed in production and therefore it was never installed in the production-based Dockerfile to keep the Docker container as slim as possible.

The only thing left after building the docker containers is to ship the production-built Docker-container to the cloud computers. This was done using the AWS Elastic Container Registry (ECR). AWS ECR simplified the process of letting the team build the containers on their local machines, and then push them over to a registry that was hosted by AWS. When this push was completed, ECR automatically handled the new container and deployed it to the cloud computing instances.

## 5.2 Second Iteration

### 5.2.1 Design 2.0

The purpose of the second iteration was to enrich both the UX and the design as well as the content. In order to achieve and fulfill the purpose, a brand new design was implemented (see Figure 5.3). The design had features that would enrich the content and make the UX a lot more comprehensive and understandable. Along with the development of the new design, Forza provided more real-time data which made it possible to provide content that is useful for the users of the web application.

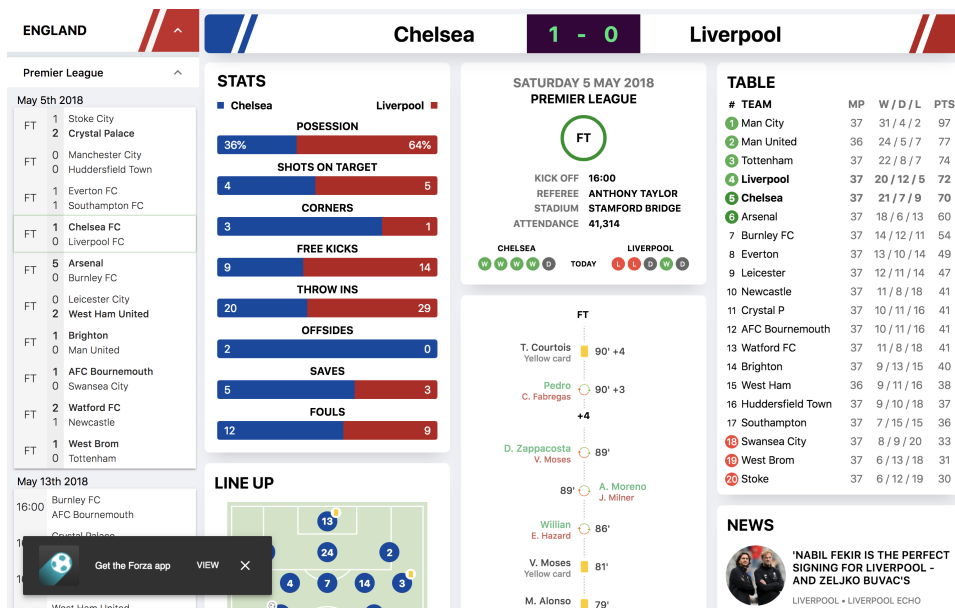
In the new design, it was decided to keep the fundamental layout with a sidebar and a dominating main component. However, the inner layout of the main component was changed. Instead of showing specific content depending on the state of a navbar it was decided to show all of the content at the same time. This put a demand on a well thought through organization of said content as not to clutter the interface. This was achieved by implementing a card layout where the available data for an entity was grouped into separate cards with the same shape and graphical profile. These cards were put into a grid following the Grid of Equals design pattern with one exception, namely that the cards were positioned in a predefined way for the specific entity. For example at a match page, the time and match events were put in the top middle section since these are the most central aspects of a match. Furthermore, if the data for a card would be unavailable the card would simply not be rendered on the interface and thus make sure that the content is relevant at any given time. This was also an advantage from a developer's perspective since it made the code more modular. For example, if new content has to be added in the future one does only have to implement a new card to add in the grid. This choice of design in the main component also allowed for a more responsive design in terms of different screen sizes. With a more narrow screen, the cards could be rearranged into columns of one or two instead of the default three.

The sidebar received a more complex structure with the introduction of additional

countries and leagues. To keep the sidebar tidy and relevant for each individual user every country were given an expandable panel containing its different tournaments. A tournament, in turn, was also given an expandable panel which if clicked would reveal its corresponding matches in a list. The matches were grouped and sorted by date and time the same way they are in the mobile application. Every match item in a list contained the most basic information depending on its status. A game that had not started yet would show its kickoff time while a live or already ended game would display the current minute and score. In addition, live matches were given a more colorful design to make it obvious which matches that were currently being played.

Because of the access to more real-time data and not having to depend on static data, as in the first iteration, it was also possible to track even more of the user behavior and flow. Data such as this is crucial in order to decide whether the UX and the design are of good quality standard.

**Figure 5.3:** The second iteration of the web application



### 5.2.2 Fetching live data

Since the big difference in the preconditions between the first and the second iteration was the access of actual live data, a lot of the time was spent on converting Forza Football's REST-API to the way the project required it to be. This was made using a separate GraphQL-server which was hosted on a separate AWS Elastic Beanstalk-instance than the actual web application. The usage of this structure made it simple to deploy updates to either the GraphQL-server or the web application, without having the other instance affected in any way.

Since GraphQL is strictly typed, it was also necessary to define all types that were

needed for the project to work. Using these type definitions, it was possible for GraphQL to check that all the types looked correct, and then create an executable schema. The only thing that was left to do was to actually specify what kind of data the different types should return using resolvers. Since the purpose of the GraphQL-server was to wrap up the already existing REST-API, all resolvers should fetch data from the REST-API, then convert the data to the desired types, and then return the result.

### 5.2.3 Optimizing the GraphQL-wrapper by caching

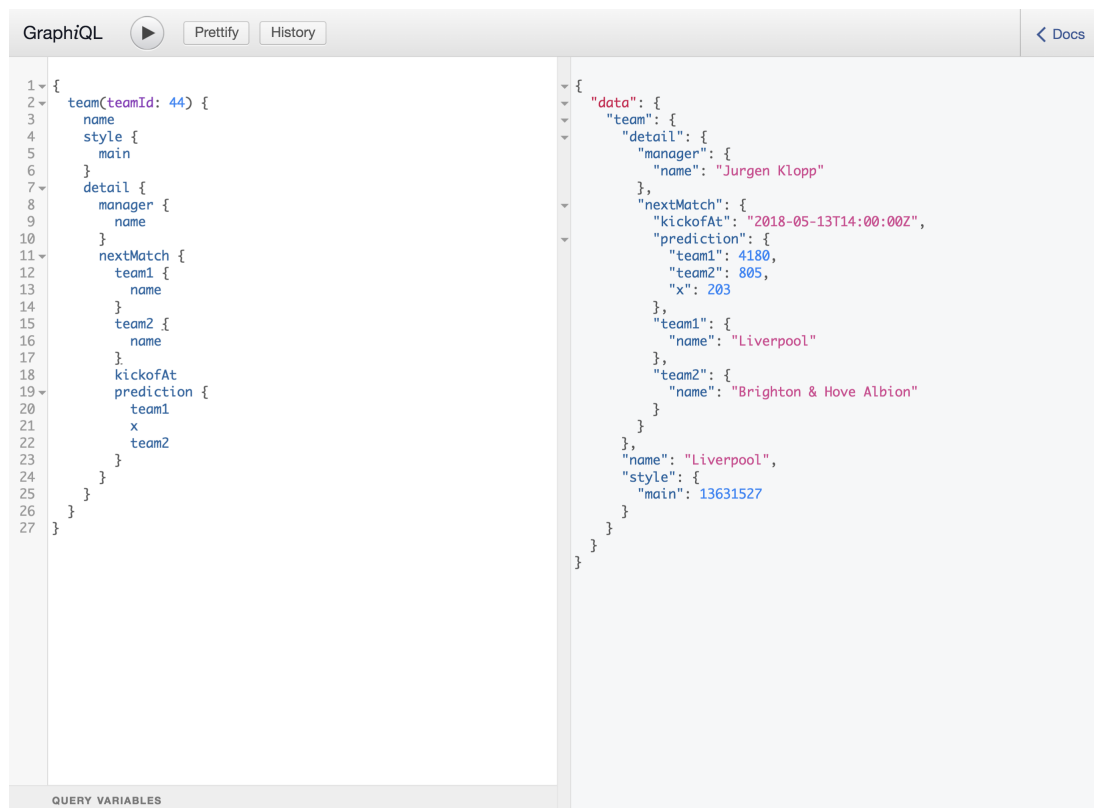
To reduce the payload for the GraphQL-server used in the project, an important step for the project was to cache everything that possibly could be cached. Not only would this method facilitate the hosted servers, it would also result in reduced response times, which in the end would end up in a more pleasant user experience. The project used two types of caching, for two different time lengths.

For objects that would be cached long term, the group built its own layer of CPU caching[43] placed in the middle of the schema and the resolvers, that could control every request that was made to the server. If the requested data existed in the cache, the data was immediately returned from the cache. However, if it could not be found in the cache, the request was instead forwarded to the resolvers. Before the resolvers return the data, it stores the newly fetched data in the cache with an expiring time. An example of where this is implemented in the project is found when requesting team names from the GraphQL-server. Since teams don't tend to change names especially often, the server is storing all possible team names that are included in the project for seven days in its cache, which reduces the number of requests made to the Forza API massively. The disadvantages of using this method are that the cache stores in the CPU, which makes the storage limited. Ideally, the CPU memory should be kept as empty as possible, for maximal server performance.

Objects with a shorter expiring time were cached using a tool called Apollo Engine, that comes bundled together with Apollo Server. Apollo Engine is a proxy that wraps up the GraphQL-server and tracks all the incoming and outgoing traffic to the server. With this data, Apollo Engine is then capable of tracing and monitoring the performance of every query that's made to the server. It's also possible to track errors that the server send when something didn't go as expected. Apollo Engine also includes Apollo Cache Control, which makes it really simple to setup and configure caching for any queries in Apollo Server. This was used to cache data that was going to be updated frequently. An example of this in the project was the news, which was cached by Apollo Engine for an hour since it's not necessary to update the news for a shorter period than that.



**Figure 5.4:** An example query run in GraphQL, fetching the team Liverpool and its next match



The screenshot shows the GraphQL IDE interface. On the left, a query is written in a monospaced font with line numbers 1 through 27. The query is a nested selection set for a team with ID 44, requesting fields like name, style, detail, manager, nextMatch, and prediction. On the right, the JSON response is displayed, showing the data for Liverpool and its next match against Brighton & Hove Albion. The response is a JSON object with a 'data' field containing the team details and match information. The team name is 'Liverpool', the manager is 'Jurgen Klopp', and the next match is scheduled for 2018-05-13T14:00:00Z.

```
1 {
2   team(teamId: 44) {
3     name
4     style {
5       main
6     }
7     detail {
8       manager {
9         name
10      }
11     nextMatch {
12       team1 {
13         name
14       }
15       team2 {
16         name
17       }
18       kickoffAt
19       prediction {
20         team1
21         x
22         team2
23       }
24     }
25   }
26 }
27 }
```

```
{
  "data": {
    "team": {
      "detail": {
        "manager": {
          "name": "Jurgen Klopp"
        },
        "nextMatch": {
          "kickoffAt": "2018-05-13T14:00:00Z",
          "prediction": {
            "team1": 4180,
            "team2": 805,
            "x": 203
          },
          "team1": {
            "name": "Liverpool"
          },
          "team2": {
            "name": "Brighton & Hove Albion"
          }
        },
        "name": "Liverpool",
        "style": {
          "main": 13631527
        }
      }
    }
  }
}
```

QUERY VARIABLES

# 6

## Result

In this chapter the final product will be presented along with statistics concerning its performance on Google, the users' behaviour on the web application and whether or not it converted users to the mobile app.

### 6.1 User Interface

The final version of the web application uses a card-based layout (see Figure 5.3), utilizing the design patterns *Center Stage* and *Grid of Equals*. The adoption of cards also encourages full use of the horizontal space. The layout is responsive in terms of screen size meaning that when visited on a more narrow web browser or on mobile the layout will change accordingly and the number of columns in the grid will be set to one, two or three depending on the width. When one column is used, most likely when the web application is used on a phone, the design pattern *Vertical Stack* is consequently utilized.

The usage of colors can be seen throughout the whole application, both the sidebar and the main view uses colors in order to further visualize and differentiate countries and teams. This enables the usage of a common, underlying, template in between pages but also the means to distinguish said pages from each other.

The application's design is centered around four views; match, team, player, and tournament. Each of these views is rendered from individual templates but have a shared sidebar. The data populated in the various templates depends on the specific entity requested.

#### 6.1.1 Sidebar

The sidebar is an essential component of the web application since it is the only global navigation and the only element shared between multiple pages. The final sidebar is based on an expandable structure where each expandable item represents a country (see figure 6.1) . Inside each expandable country item, all the leagues in that specific country are represented as another level of expandable items. The decision for this is based on the fact that in the future there will be over 180 countries with at least one league being represented per each country and the only implementation found eligible for this matter is by using expansion panels according to Google's Material UI Design Guidelines[44]. The countries are ranked top to bottom in the

order of the interest for the country in question.

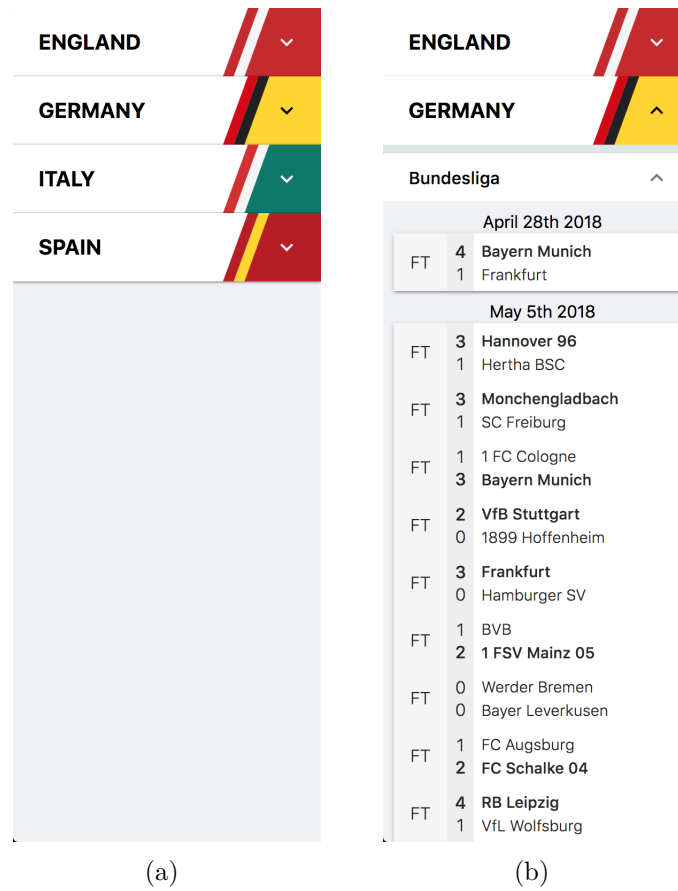
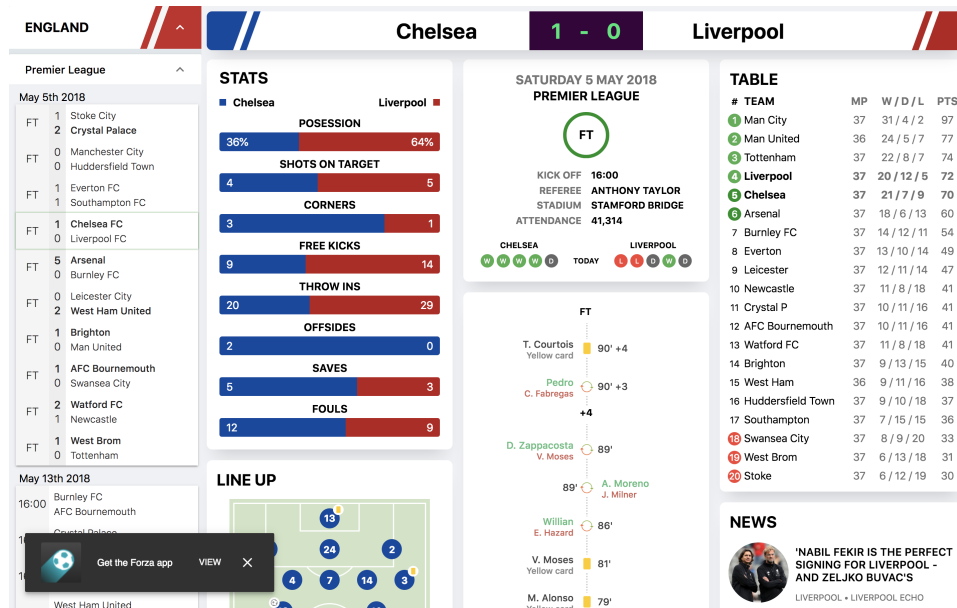


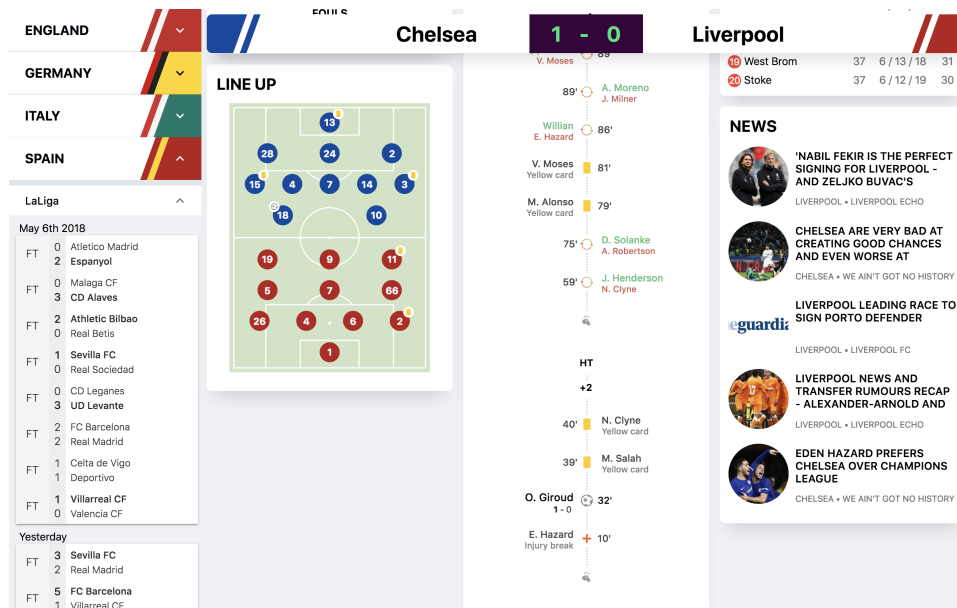
Figure 6.1: Sidebar in expanded/non-expanded state

### 6.1.2 Match view

Below, an example of the match page on desktop can be seen (figure 6.2).



(a)



(b)

Figure 6.2: Example of a Match view - Desktop

In a match, the current score is of most and first interest of many visitors. That is why an almost straight line with information, originating from the score up top to the bottom, can be identified. This is where the most important information is located, such as score, date, match clock and lastly a timeline with the match events. Moreover, the column to the left as well as the column to the right contains more information relevant to the given match. The cards "Stats" and "Line up" are placed in the leftmost column, which utilizes a team's color to separate the information. The rightmost column displays information that might not be directly related to a match but still relevant and interesting. In the final application, the "Table" and

"News" card is located here.

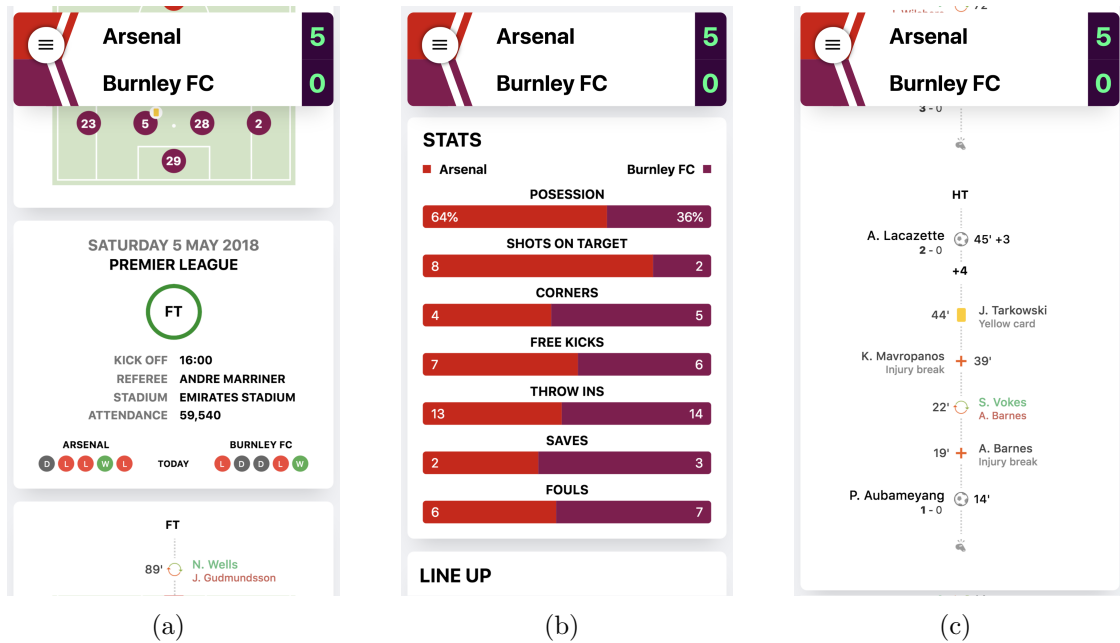


Figure 6.3: Example of a Match view - Mobile

An example of the match view in mobile mode can be seen in figure 6.3. Instead of 3 columns, all cards are stack upon each other in a single column.

### 6.1.3 Team view

Figure 6.4 shows a screen capture of a team's page from a desktop and figure 6.5 on mobile.

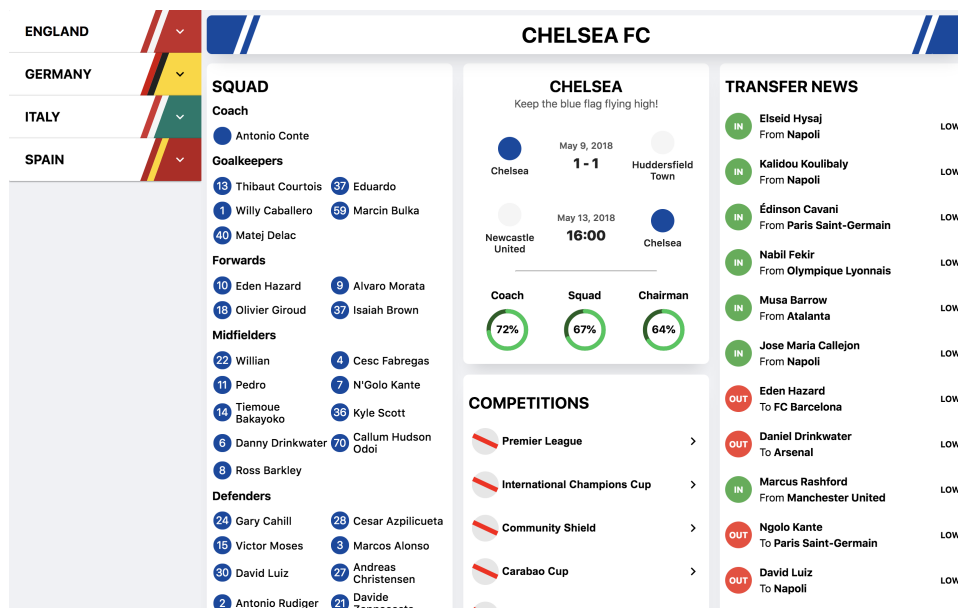


Figure 6.4: Example of a Team view - Desktop

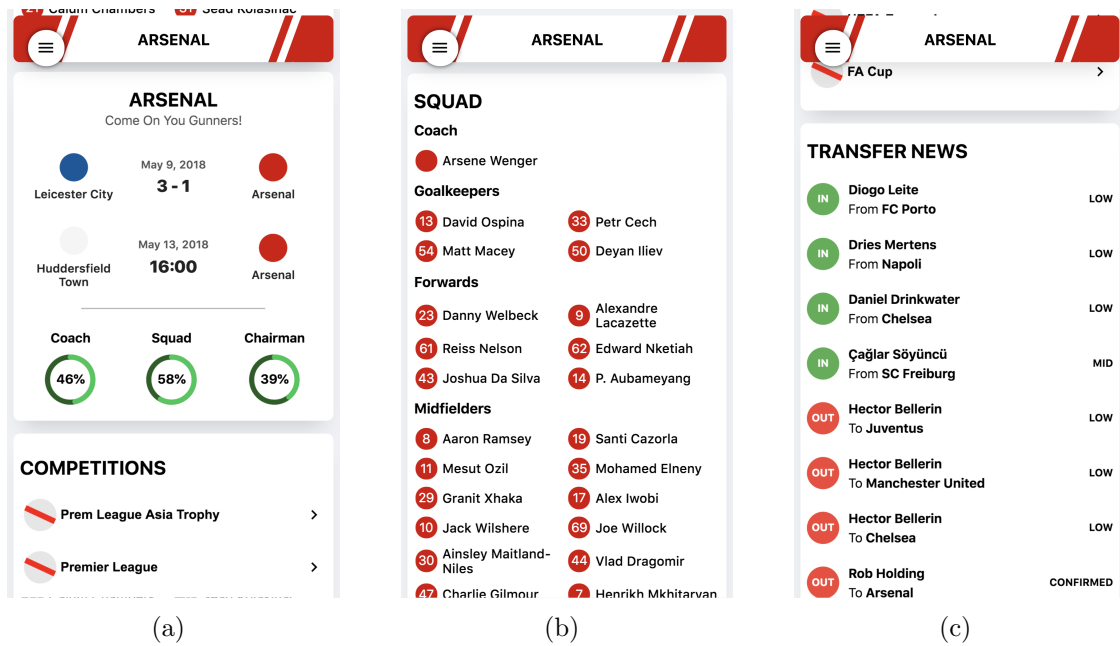


Figure 6.5: Example of a Team view - Mobile

The cards featured on the team page have taken much inspiration from the Forza application. Previous and next game is at the center of attention. A team's squad and transfer news are examples of more data that is visualized here.

## 6.2 Visibility and Incoming Traffic

This section will cover how the site did in terms of rank, exposures and clicks on Google. All queries have been done in Google Chrome's Incognito mode since this will make the search algorithm ignore one's previous internet behavior and not favor football related hits due to earlier searches.

### 6.2.1 Did it rank highly for core themes and target phrases?

Before implementing the changes to improve SEO in the first iteration the web application's presence in Google's search results were close to non-existent. After implementing the first techniques, immediate improvement could be seen on the site's ranking. Search queries would then generate hits for the web application on the first few pages where they previously did not appear at all. As of now, the web application's rank on core themes and relevant target phrases varied depending on league and what words that were included in the queries. For example, the inclusion of the word 'livescore' would often boost the rank several pages when searching for matches. A selection of results from relevant search queries on Google has been compiled below. The queries have been concentrated on three of the most popular leagues in the world and include queries for big and small teams both, in terms of popularity.

A Google search for 'Villarreal Celta livescore', a Spanish league game, two days after it had been played, resulted in the web application appearing on the top of the fourth page. If excluding 'livescore' the web application dropped out of the first twenty pages. However, omitting the same keyword when searching for 'Valencia vs SD Eibar' generated a hit on the very first page of Google. But in general, matches including more popular teams tended to rank lower than matches between teams that might not get the same international media exposure. This pattern was prominent in the other leagues as well.

Queries, La Liga		
Search term	Page	Position
Villarreal Celta Vigo livescore	4	1
Valencia vs SD Eibar	1	10
Malaga livescore	3	3

**Figure 6.6:** Queries, La Liga (full size screen shots exist in appendix A)

A search for 'Bayern Munich livescore', Bayern Munich being the most popular team in the German league, resulted in the corresponding team page appearing on page three. When omitting 'livescore' the web application is not included in the hits at all. If searching for the recently played game between Freiburg and Cologne along with 'livescore' the match page did not show up anywhere in the first ten pages. However, the team page for Freiburg appeared already on the first page suggests that even though the web application tended to rank high with the right queries it might not always be the exact content the user was looking for. This result can also be seen with the game 'Frankfurt vs Hamburger SV', the team page for Frankfurt is the only result that is shown.

Queries, Bundesliga		
Search term	Page	Position
Bayern Munich livescore	3	4
Freiburg Cologne livescore	1	7
Frankfurt Hamburger SV Livescore	1	9

**Figure 6.7:** Queries, Bundesliga (full size screen shots exist in appendix A)

The same patterns as for the other leagues could also be seen when searching for matches in the English Premier League. A search for the match between Manchester United and Arsenal, two of the most popular teams in the world[45], did not result in any hits for the web application on any of the first ten pages, even when including 'livescore'. In contrast, both 'Crystal Palace Leicester livescore' and 'Southampton Bournemouth livescore' generated a hit on the second page. Also, with the phrase 'Tottenham livescore' the corresponding team page was found on the fourth page.

Queries, Premier League		
Search term	Page	Position
Crystal Palace Leicester livescore	2	6
Southampton Bournemouth livescore	2	1
Tottenham livescore	4	1

**Figure 6.8:** Queries, Premier League (full size screen shots exist in appendix A)

The pattern of smaller teams, in terms of popularity, getting better exposure on the hit lists can also be seen in the data collected in the Google Search Console. When looking at the top 10 most used search queries for entering the web application during April seven out of ten contains teams not considered being among the top 3 most popular ones in their respective countries.

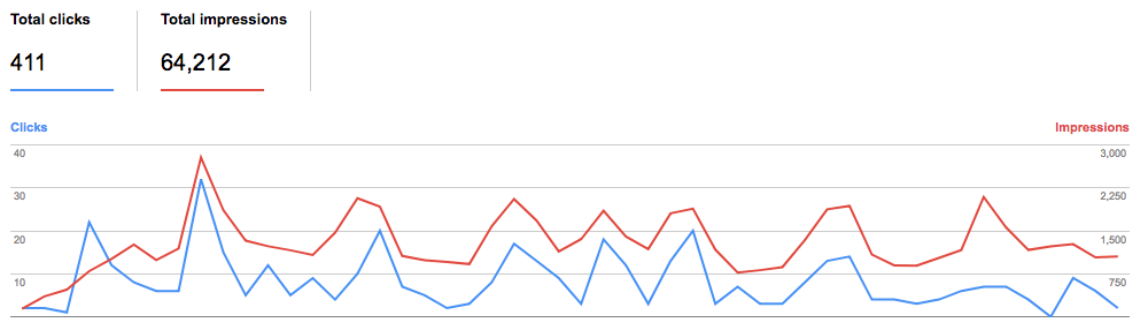
Queries	Clicks ▼	Impressions	Position
1 girona fc livescore	3	319	9.1
2 ud levante vs las palmas ud	2	30	9.7
3 lorenzo bruzzi	2	3	9.0
4 valencia vs getafe	2	12	46.8
5 emanuele torrasi	2	7	21.0
6 augsburg livescore	2	129	10.3
7 schalke 04 livescore	2	354	9.3
8 lazio roma livescore	2	41	9.4
9 malaga vs sociedad	2	3	16.3
10 huddersfield vs everton livescore	2	38	8.8

**Figure 6.9:** Google search console, Top 10 search queries before entering the web application, April 2018

## 6.2.2 How much traffic was driven to the site?

Looking in the Google Search Console there are two units of measurement to take into account when reviewing this question. Foremost there is the total amount of clicks that takes a visitor from Google's search result list to the web application. During the (almost) two months that the web application has been monitored in the Search Console it has been entered from the list of search results 411 times. This can be put into relation to the number of impressions that the web application have gotten, an impression being that the web application is included in the page of the results that the user currently is watching. This number amounts to 64,212 giving the web application a click frequency of 0,64% when included on the page. It should be noted that even though the user has not scrolled down far enough in their browser for the web application to actually be visible to them it still counts as one impression.

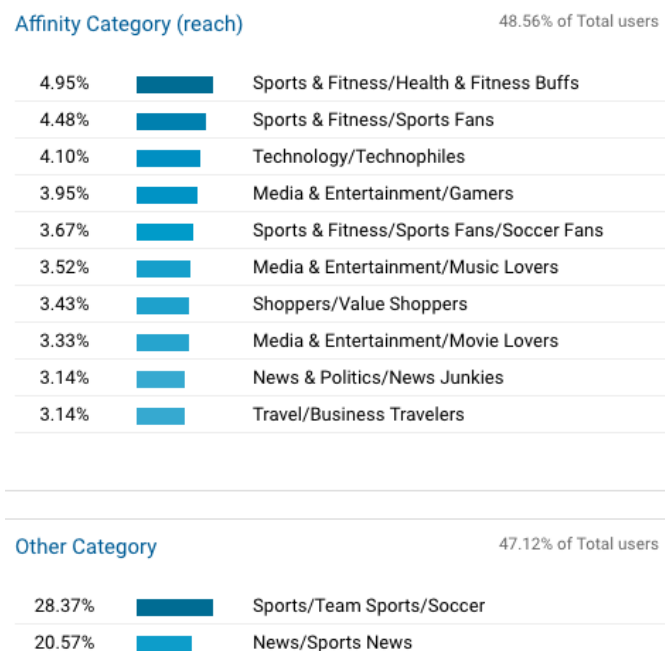




**Figure 6.10:** Google Search Console, clicks(blue) and impressions(red) for the web application, 23/03/18 - 11/05/18

### 6.2.3 Did it attract the target audience?

Even if it's vital to have a large number of visitors, it's equally important that they have the appropriate interests. Search engines today measures the time away from the search page, after clicking an outgoing link. Thus, attracting the wrong target group could affect the page rank negatively if visitors leave the page quickly since that would indicate that the content is irrelevant. In order to get an overview of the interests of the visitors, data collected in Google Analytics was compiled into a list.



**Figure 6.11:** Google Analytics, Interest categories of audience, 26/04/18 - 11/05/18

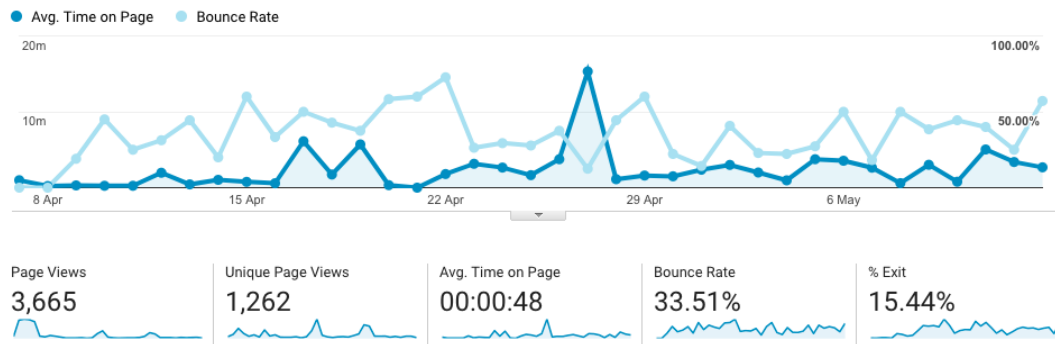
As one can see from figure 6.11, the majority of the incoming traffic has sports-related interest. This entails that the web application is successful in attracting the target audience.

## 6.3 Content and User Interactions

This section will go into detail concerning how the visitors interacted with the content of the web application, how much time they spent on it and if it eventually leads to downloads of the mobile application.

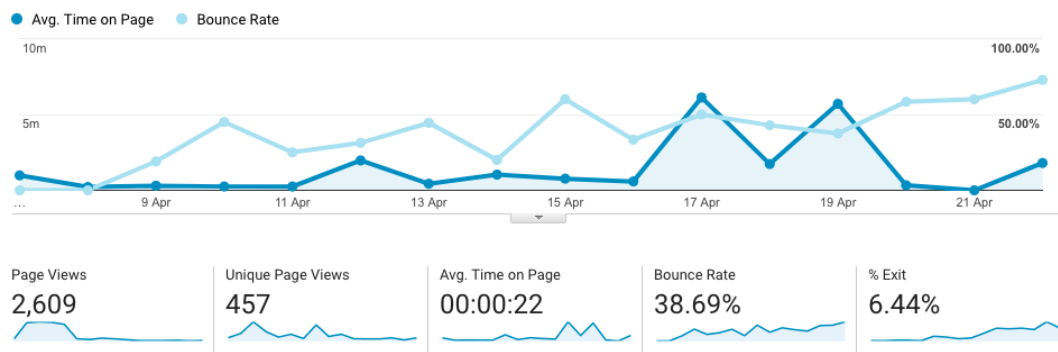
### 6.3.1 Did the visitors bounce or did they find the content engaging and interacted with it?

To answer these two questions, data collected from Google Analytics have been compiled into a diagram showing the average time spent on the site in relation to the bounce rate. This data shows that the average time spent on a page is 48 seconds and that the bounce rate is about 34%. These numbers cover the span 07/04/18-13/05/18.

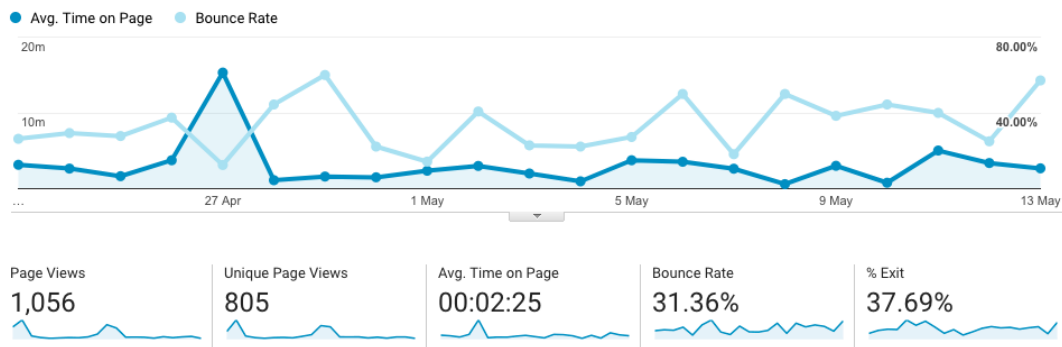


**Figure 6.12:** Average time spent on page vs Bounce rate

When looking at the data for the time spans for the first and second design separately there are some differences worth highlighting. After the launch of the new design at 23/04/18 the visitors stayed, in average, about 2 minutes longer on the site every session, or if put into relation to the data for Design 1.0; an increase with 559%. Furthermore, the bounce rate dropped 7,33 percentage units. These numbers indicate that the visitors find the content more valuable and engaging with the new design.



(a) Design 1.0

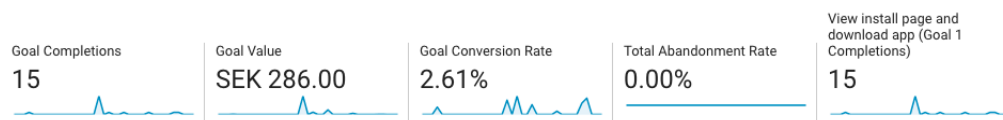


(b) Design 2.0

**Figure 6.13:** Average time spent on page vs. Bounce rate comparison

### 6.3.2 Did the content fuel goal completions/conversions or drive sales?

At the moment of this report being finalized the number of visitors that have ended up going to App Store/Google Play is 13 with the conversion rate being 2,61%. There is no data available on how many of the visitors that went to App Store/-Google Play that actually downloaded the application. This is due to the use of the wrong campaign, a tool for measuring where visitors origin from.

**Figure 6.14:** Goal completion, value and conversion rate for the web application

### 6.3.3 Did the web application perform well enough?

Using the open source test called Lighthouse[46] makes it possible to audit the performance of the web application. Since Lighthouse originally was created by Google, running this test will probably give an insight of how Google thinks the web application performs as well.



**Figure 6.15:** A generated Lighthouse-test

Based on this test, the performance of the web application was fairly good, even though there's still room for improvement. However, the fact that the web application got a score of 100 on the SEO-audit was a confirmation that the implementation of the guidelines from Google was successful.

# 7

## Discussion

This section aims to discuss the results of the web application, what could be improved and evaluate the goals that were brought up in the introduction.

### 7.1 User Interface

The final result of the user interface solved the main potential issues thought of at the beginning of the project, namely how to design a web application of this kind that is responsive, detailed yet not cluttery, as well as visually appealing. Due to the fact that the available data varies depending on whether the match has been played or not, the card layout is suitable since the card is not displayed if there is no accompanying data. One of the things that can be further improved with the design is to make the positioning of the cards more dynamic so that the structure of the cards is good looking independent of how many cards there are. A solution to this problem was attempted, but the available libraries to solve this problem were slow and a self-made solution would have taken too much time to implement. Nevertheless, the positioning has other flaws that could be improved such as the ordering of the cards when the screen size is decreased. Additionally, some of the guidelines from Google have not been fulfilled. One of them is font size; currently, the web application has 52% of the text in size 16 pixels or larger, although Google recommends having at least 75% to increase readability on mobile devices. The reason for not fulfilling this guideline is that the minimum card width limits the use of a consistent font size in favor of a better-looking card. Another guideline that Google complains about when analyzing the web application is color contrast. A possible solution to this is a better algorithm to decide which font color that suits the different background colors of the elements. Also, the HTML-tag structure may have flaws which are not optimal when Google indexes sites or for example when using screen reader plugins. Finally, something that has been worked on but still has room for improvement is the sidebar and how it is accessed on different screen sizes.

### 7.2 Visibility

The following goals were stated in the introduction:

1. Web application appears on at least the 2nd page when searching for a random match on Google

2. Web application appears on at least the 3rd page when searching for a random team on Google
3. Web application appears on at least the 3rd page when searching for a random player on Google
4. Web application appears on at least the 4th page when searching for a random tournament on Google
5. 3% of the visitors of the page go to the App Store page for the Forza application.
6. 25% that goes to App Store downloads the Forza application.

Goals 1-4 are very similar and they will be bunched together as the answer is the same for all of them. For less popular (less searched for) teams/players/matches the web application successfully appears early. For much more popular (more searched for) teams/players/matches the web application will often end up on the first 5 pages. The goals were set as searching for a random term (match, team, player, tournament) would make the web application appear early and this was not fully achieved. Searching with the term and the word *livescore* is important for appearing early in a search result. All goals in 1-4 combined with the search query 'term + livescore' were successfully achieved.

Goal 5 was almost met, the actual percentage was 2,61% in comparison to the 3% desired. However there is some doubt about how many of the visitors that went to App Store/Google Play were part of the developers. This could have been fixed by filtering the developers IP Address from the tracking server, however, this was thought of too late to matter.

Goal 6 is hard to measure since the wrong campaign (tool for measuring where the visitor came from) was used. The data from Forza is often changed but the current statistics from April showed that 8% of the visitors to Google Play downloaded their application. How many of these visitors that came from the web application is a very small number since the web application only had 411 visitors as seen in figure 6.10 and they had over 650,000 downloads during the whole projects duration.

## 7.3 Handling the Data

The result of how the application handles the data do answer the problem question that was defined in the Problem Statement: *How does this data gets converted to a fully working web application?* Based on the provided data given from Forza, the application successfully manage to generate dynamic web pages with dynamic URL routes. However, the flow of the provided data changed during the development of the project from first using SQL-dumps only using the data through the live API. This was because late in the project it was announced that there will be no more SQL-dumps available and that all data should come from the live API. There was no time to reconstruct the project using no SQL-dumps, which resulted in a limited amount of pages generated. Since the web application's SEO performed best with

matches and teams that don't get as much international media coverage as the most popular ones, the application would probably have received a better SEO result if it would cover all the matches and teams that Forza does in their mobile application.

# 8

## Conclusion

The main purpose of this thesis was to investigate how a web application could be utilized to acquire new users to a mobile application. In order to do this, a web application was developed in collaboration with Forza Football with the main objective of generating downloads of their main mobile application. The web application was live on `livescore.forzafootball.com` after two months with an initial design, which was replaced a month later with a revised design. The completed product contains four leagues (Premier League, La Liga, Bundesliga and Serie A) with all its matches, teams and players.

The web application had in May over 60,000 impressions and over 400 clicks. The main interest of the visitors was sport/football. 2,61 % of the visitors went to App Store/Google Play, which almost fulfills the fourth goal stated at the beginning of the project. No data was obtained on the number of downloads of the Forza application as an outcome of the web application, which made it difficult to evaluate that particular goal. The remaining goals were about what rank the site had for different search queries. The web application generally appeared as one of the top results when searching for a football term together with the word *livescore*, but not as high when searching for teams or matches only. Considering how late the live data was obtained, the overall result is gratifying and indicates potential to reach better numbers eventually. The project shows that the most recommended SEO techniques in order to increase the visibility of a brand on search engines are probably an effective way of acquiring customers.



# Bibliography

- [1] Forza Football, 2018 [Online] Available: <https://www.forzafootball.com/> [Accessed: May 14, 2018]
- [2] We are social, 2018 Q2 Global Digital Statshot, April 17, 2018. [Online] Available: <https://www.slideshare.net/wearesocialsg/2018-q2-global-digital-statshot-94084375> [Accessed: May 12, 2018].
- [3] Molly Galetto, NgData. *What Is Customer Acquisition?*. May 2, 2018. [Online] Available: <https://www.ngdata.com/what-is-customer-acquisition/> [Accessed: May 12, 2018].
- [4] Zac Gregg, Vital. *Inbound Marketing vs. Outbound Marketing* [Online] Available: <https://vtldesign.com/digital-marketing/inbound-marketing-vs-outbound-marketing/> [Accessed: May 12, 2018].
- [5] World Wide Web Foundation. *Internet Live Stats* [Online] Available: <http://www.internetlivestats.com/> [Accessed: May 12, 2018].
- [6] Wikipedia. *Search engine optimization*. April 23, 2018 [Online] Available: [https://en.wikipedia.org/wiki/Search\\_engine\\_optimization](https://en.wikipedia.org/wiki/Search_engine_optimization) [Accessed: May 12, 2018].
- [7] Haris Basic. *Become a data genius, track everything in your small business*. June 20, 2016 [Online] Available: <https://www.forbes.com/sites/allbusiness/2016/06/20/become-a-data-genius-track-everything-in-your-small-business/#63732b1f8797> [Accessed: May 12, 2018].
- [8] StatCounter Global Stats. *Search Engine Market Share Worldwide*. [Online] Available: <http://gs.statcounter.com/search-engine-market-share> [Accessed: May 12, 2018].
- [9] Aravind Shenoy and Anirudh Prabhu, *Introducing SEO*. Apress, 2016 [Book]
- [10] Alphabet. *Alphabet Quarterly Report*. 2018-10-Q. [Online] Available: [https://abc.xyz/investor/pdf/20180423\\_alphabet\\_10Q.pdf](https://abc.xyz/investor/pdf/20180423_alphabet_10Q.pdf) [Accessed: May 14, 2018].
- [11] Google. *Bounce rate*. [Online] Available: <https://support.google.com/analytics/answer/1009409?hl=en> [Accessed: May 13, 2018].
- [12] Sergey Brin and Lawrence Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. August 1998. [Online] Available: <http://ilpubs.stanford.edu:8090/361/1/1998-8.pdf> [Accessed: May 13, 2018].
- [13] W3Schools. *HTML <meta> Tag*. [Online] Available: [https://www.w3schools.com/tags/tag\\_meta.asp](https://www.w3schools.com/tags/tag_meta.asp) [Accessed: May 13, 2018].
- [14] Moz. *Meta Description* [Online] Available: <https://moz.com/learn/seo/meta-description> [Accessed: May 13, 2018].

- [15] Moz. *Title Tag* [Online] Available: <https://moz.com/learn/seo/title-tag> [Accessed: May 13, 2018].
- [16] Google. *Using site speed in web search ranking*. April 9, 2010. [Online] Available: <https://webmasters.googleblog.com/2010/04/using-site-speed-in-web-search-ranking.html> [Accessed: May 13, 2018].
- [17] Google. *HTTPS as a ranking signal*. August 6, 2014. [Online] Available: [https://security.googleblog.com/2014/08/https-as-ranking-signal\\_6.html](https://security.googleblog.com/2014/08/https-as-ranking-signal_6.html) [Accessed: May 13, 2018].
- [18] Google. *Learn about sitemaps*. [Online] Available: <https://support.google.com/webmasters/answer/156184> [Accessed: May 13, 2018].
- [19] Google. *Responsive Web Design*. September 27, 2017. [Online] Available: <https://developers.google.com/search/mobile-sites/mobile-seo/responsive-design> [Accessed: May 13, 2018].
- [20] Wikipedia. *User experience*. May 9, 2018 [Online] Available: [https://en.wikipedia.org/wiki/User\\_experience](https://en.wikipedia.org/wiki/User_experience) [Accessed: May 12, 2018].
- [21] All about UX. *User experience definitions*. March 31, 2018 [Online] Available: <http://www.allaboutux.org/ux-definitions> [Accessed: May 12, 2018].
- [22] Jakob Nielsen. *Minimize Cognitive Load to Maximize Usability*. December 22, 2013 [Online] Available: <https://www.nngroup.com/articles/minimize-cognitive-load/> [Accessed: May 12, 2018].
- [23] Jennifer Tidwell, *Designing Interfaces*. O'Reilly Media, 2011 [Book]
- [24] Nick Babich. *Basic patterns for mobile navigation*. September 8, 2016. [Online] Available: <https://uxplanet.org/basic-patterns-for-mobile-navigation-d12a87686efe> [Accessed: May 13, 2018].
- [25] Kate Meyer. *How Chunking Helps Content Processing*. March 20, 2016. [Online] Available: <https://www.nngroup.com/articles/chunking/> [Accessed: May 13, 2018].
- [26] Jakob Nielsen. *Mental Models*. October 18, 2010. [Online] Available: <https://www.nngroup.com/articles/mental-models/> [Accessed: May 13, 2018].
- [27] Jakob Nielsen. *Website Response Times*. June 21, 2010. [Online] Available: <https://www.nngroup.com/articles/website-response-times/> [Accessed: May 13, 2018].
- [28] Chris Horsnell, Medium. *"What is Agile Workflow? (Eli5)?"*. September 18, 2017. [Online] Available: <https://medium.com/scrumi/what-is-agile-workflow-eli5-15040cbd5e75> [Accessed: May 12, 2018].
- [29] React. [Online] Available: <https://reactjs.org/> [Accessed: May 13, 2018].
- [30] Introduction to the DOM [Online] Available: [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model/Introduction](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction) [Accessed: April 2, 2018].
- [31] Zeit. [Online] Available: <https://zeit.co/about> [Accessed: May 13, 2018].
- [32] Wikipedia. *Node.js* [Online] Available: <https://en.wikipedia.org/wiki/Node.js> [Accessed: May 14, 2018].
- [33] Google. *Analytics*. [Online] Available: <https://analytics.google.com/analytics/web/> [Accessed: May 14, 2018].

- 
- [34] MongoDB. *MongoDB and MySQL Compared*. [Online] Available: <https://www.mongodb.com/compare/mongodb-mysql> [Accessed: May 14, 2018]
  - [35] Wikipedia. *Amazon Web Services*. May 9, 2018. [Online] Available: [https://en.wikipedia.org/wiki/Amazon\\_Web\\_Services](https://en.wikipedia.org/wiki/Amazon_Web_Services) [Accessed: May 13, 2018].
  - [36] AWS. *Cloud Computing with Amazon Web Services*. [Online] Available: <https://aws.amazon.com/what-is-aws/> [Accessed: May 13, 2018].
  - [37] Synergy Research Group. *Cloud Growth Rate Increased Again in Q1; Amazon Maintains Market Share Dominance*. April 27, 2018. [Online] Available: <https://www.srgresearch.com/articles/cloud-growth-rate-increased-again-q1-amazon-maintains-market-share-dominance> [Accessed: May 13, 2018].
  - [38] GlobeNewsWire. *2nd Watch Identifies the Most Popular AWS Products of 2017*. February 1, 2018. [Online] Available: <https://globenewswire.com/news-release/2018/02/01/1329867/0/en/2nd-Watch-Identifies-the-Most-Popular-AWS-Products-of-2017.html> [Accessed: May 13, 2018].
  - [39] Brian Douglas. *Service Workers Explained*. October 31, 2017. [Online] Available: <https://www.netlify.com/blog/2017/10/31/service-workers-explained/> [Accessed: May 13, 2018].
  - [40] Wikipedia. *Representational state transfer*. May 12, 2018. [Online] Available: [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer) [Accessed: May 13, 2018].
  - [41] Google, May 2018 [<https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/webfont-optimization>].
  - [42] Docker. [Online] Available: [https://hub.docker.com/\\_/node/](https://hub.docker.com/_/node/) [Accessed: May 13, 2018].
  - [43] Wikipedia. *CPU cache*. May 7, 2018. [Online] Available: [https://en.wikipedia.org/wiki/CPU\\_cache](https://en.wikipedia.org/wiki/CPU_cache) [Accessed: May 13, 2018].
  - [44] Google Material IO. *Expansion panels*. [Online] Available: <https://material.io/archive/guidelines/components/expansion-panels.html#> [Accessed: May 14, 2018].
  - [45] Kurt Badenhause. *FC Barcelona ranks as the top sports team on social media*. July 14, 2016 [Online] Available: <https://www.forbes.com/sites/kurtbadenhausen/2016/07/14/fc-barcelona-ranks-as-the-top-sports-team-on-social-media/#41dc45b06ab1> [Accessed: May 13, 2018].
  - [46] Google. *Lighthouse*. April 9, 2018. [Online] Available: <https://developers.google.com/web/tools/lighthouse/> [Accessed: May 29, 2018].
  - [47] Tower. *Learn Version Control with Git*. [Online] Available: <https://www.git-tower.com/learn/git/ebook/en/desktop-gui/basics/why-use-version-control> [Accessed: May 12, 2018].
  - [48] Atlassian Bitbucket. *Git Feature Branch Workflow*. [Online] Available: <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow> [Accessed: May 12, 2018].
  - [49] MySQL. [Online] Available: <https://www.mysql.com/products/> [Accessed: May 13, 2018].

- [50] Wikipedia. *MySQL*. May 8, 2018. [Online] Available: <https://en.wikipedia.org/wiki/MySQL> [Accessed: May 13, 2018].
- [51] Facebook. *GraphQL: A data query language*. September 15, 2015. [Online] Available: <https://code.facebook.com/posts/1691455094417024> [Accessed: May 13, 2018].
- [52] Apollo. [Online] Available: <https://www.apollographql.com/docs/react/> [Accessed: May 13, 2018].
- [53] Docker. *What is Docker?*. [Online] Available: <https://www.docker.com/what-docker> [Accessed: May 14, 2018].
- [54] Docker. *Dockerfile reference* [Online] Available: <https://docs.docker.com/engine/reference/builder/> [Accessed: May 14, 2018].
- [55] Docker. *What is a container?*. [Online] Available: <https://www.docker.com/what-container> [Accessed: May 14, 2018].

# A

## Appendix 1

### A.1 Feature Branch

Using a Version Control System (VCS), in this case Git, is of utter importance during the process of a project since it enables seamless collaboration between team members. Several team members may work on the same file and then, using a VCS, merge together all the changes into a single updated file. Also, as the name VCS indicates, it's easy to keep track of versions so that one can always revert to an earlier version if necessary[47].

The project has been hosted on GitHub that is a web host using Git as VCS. Moreover, the workflow that's been used on GitHub is called *Git Feature Branch Workflow*. The core idea with Feature Branch is that all feature development, that is all development of new implementations, should take place in a dedicated branch and not in the *master* branch (*master* branch is the code ready for deployment). The motivation for this idea is that by doing this it encapsulates the development and makes it easy for multiple developers to work on a given feature without disturbing or interfering with the main code. Also, by following the core idea of Feature Branch the *master* branch will never contain any broken code.

Moreover, the guidelines for the Feature Branch workflow also suggests that it should be a standard to leverage pull requests. A pull request appears when some development branch is ready to go into the *master* branch, *i.e.* into production. The idea is that the person responsible for the branch opens up for reviewing and discussion by opening a pull request. This has been the core flow of the team in order to make it easy to follow other team members progress but also to give constructive feedback and input on the code being reviewed. Pull request enhances code quality and also prevents broken code being merged into *master* to a greater extent than if not used[48].

### A.2 MySQL

MySQL is an open-source Relational Database Management System (RDMS). It is the world's most popular open source database, used by companies like Google, Facebook, Twitter, PayPal and YouTube[49]. It's using the language SQL for querying the data needed. Even though the initial release was back in 1995[50], MySQL is still considered one of the most secure and reliable RDMS. It's designed to be

able to run the most demanding applications and it's trusted by frameworks like Wordpress, Joomla and Drupal.

### A.3 GraphQL

GraphQL is a relatively new data query language (initial released 2015) that is used to communicate with a server[51]. With GraphQL, a client can shape the data that the server should return when requesting data using GraphQL-queries. The advantages of this method are that the client easily can predict the shape of the data returned from a query since it's the client that specifies what it wants. This makes it easy to add new features and fields to the server since you don't have to affect any of the existing fields while creating new ones. GraphQL queries also reduce the response time from the server since it's only returning exactly what's needed and nothing else. GraphQL is strongly typed containing different entities that are available to query, and where each of these entities describes a set of available fields. This makes it possible to catch any errors in advance of execution, which gives confidence to the client since it knows that the queries are correctly written.

During this project, an implementation of GraphQL called Apollo was used. Apollo provides both Apollo Server and Apollo Client[52], services that make it easy to get started with a GraphQL-server and then seamlessly make queries to the server from any type of client. Apollo Server does also come bundled with GraphiQL, which is an IDE that makes it possible to graphically test the queries against the database. This is a good way to make sure that a client gets the desired result.

### A.4 Docker

Docker, that is developed by a company with the same name, is used to eliminate the obstacles caused by the conflicts between the applications, infrastructure and the developers together with limitations of their hardware. Docker's goal is to create a model for better collaboration and innovation[53]. Instead of installing and setting up all the required tools locally, one only needs Docker and the mutual Dockerfile to build up a development environment. A Dockerfile is a text document, customized by developers, that contains all the necessary commands to assemble all the tools which will be used by Docker to create a build for the development environment[54]. With help of the Dockerfile, Docker provides a mutual and consistent development environment on a virtual machine using the resources supplied by the development machine it runs on. It can reduce the required resources by running multiple containers on the same single virtual machine instead of starting multiple virtual machines on an operating system. In this case, a container is a stand-alone and executable package of a piece of software that contains all the tools, libraries, settings and so on required to run the software itself[55]. Docker also solves the problem that might show up when different kinds or versions of OS, development tools and so forth are in use. Furthermore, every change in the choice of versions and type of developing

tools results in the same change for other developers by updating the mutual Dockerfile and pushing to GitHub.

Since Docker has been relatively new in the industry, a lot of time was spent setting up and making sure that it worked properly for everyone. Together with GitHub, every time a change in the Dockerfile has been made, a push or pull request to GitHub will also be made so other developers on the same branch may acquire the change and sometimes even without knowing about it. In addition, knowing that these kinds of changes have been made a few times a week, it's important that Docker is moving files and installing tools in right order to optimize the building of the development environment. In short, Docker has been a great help easing the development of the web application by eliminating the time demanded everyone to adjust after a change and also to create a change in the mutual development environment.

## A.5 Figures

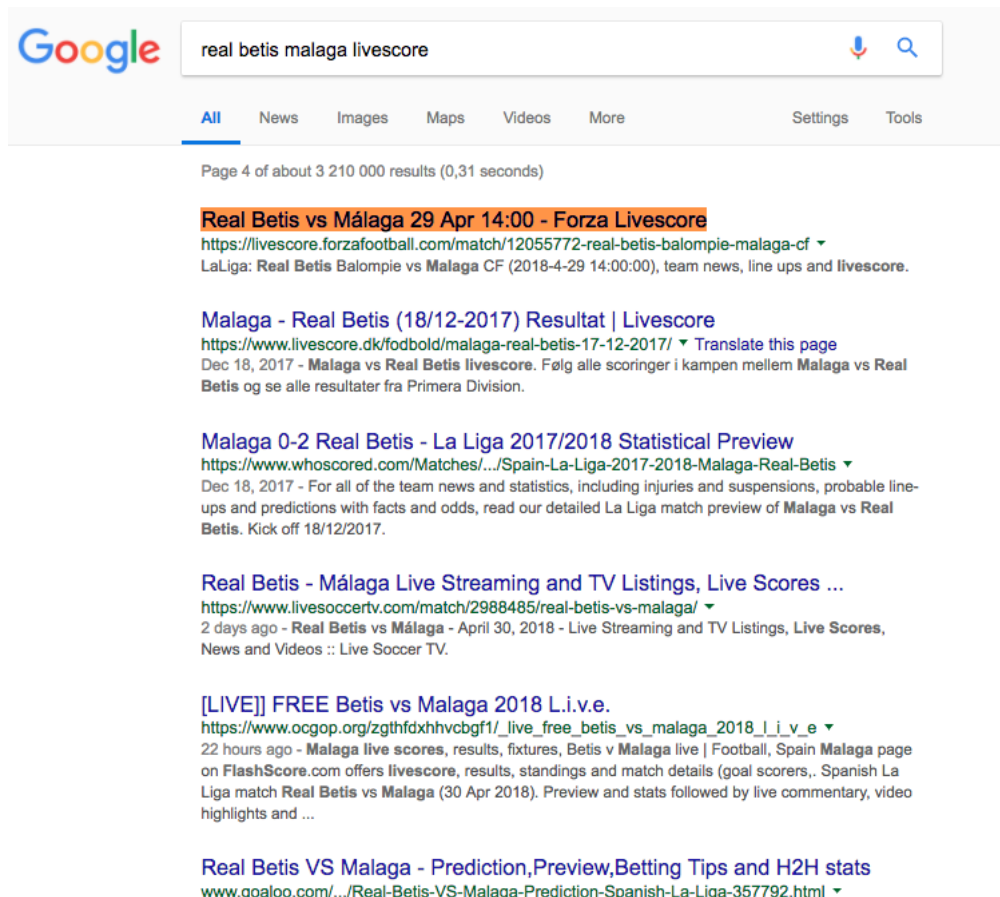


Figure A.1: Real Betis Malaga livescore. Source: 6.6

**valencia cf vs. sd eibar preview - valencia cf Oficial webpage**

[en.valenciacf.com/ver/51274/valencia-cf-vs--sd-eibar-preview.html](https://en.valenciacf.com/ver/51274/valencia-cf-vs--sd-eibar-preview.html) ▼

HISTORIC FACTS Both league games between these two sides have been won by Valencia. Valencia CF have scored four goals and conceded one against SD Eibar. Paco Alcácer scored the goal in the win of the last season at Ipurúa. The last game at Mestalla ended with a 3...

**Compare teams – Valencia CF vs SD Eibar – Futbol24**

[www.futbol24.com/teamCompare/Spain/Valencia.../vs/.../SD-Eibar/...](https://www.futbol24.com/teamCompare/Spain/Valencia.../vs/.../SD-Eibar/...) ▼ [Translate this page](#)

Compare teams – Valencia CF vs SD Eibar – Futbol24.

**Valencia vs Eibar - Predictions, Free Bets, Odds and Betting Tips ...**

<https://www.bigfreebet.com/.../valencia-vs-eibar-predictions-free-bets-odds-and-bettin...> ▼

4 days ago - Valencia CF vs SD Eibar – Betting Tips for the game. Valencia were on an amazing run of nine league games without defeat until two weeks ago. However, they are now winless in three La Liga fixtures, including a 1-1 away draw at Celta Vigo last weekend. At home, Valencia were on a five-game winning ...

**Valencia vs SD Eibar 29 Apr 14:00 - Forza Livescore**

<https://livescore.forzafootball.com/match/12055764-valencia-cf-sd-eibar> ▼

LaLiga: Valencia CF vs SD Eibar (2018-4-29 14:00:00), team news, line ups and livescore.

**Searches related to valencia vs sd eibar**

eibar vs valencia

eibar vs valencia **results**

eibar vs valencia **h2h**

eibar vs valencia **highlights**

valencia vs eibar **live**

eibar vs **barcelona**

eibar vs valencia **live stream**

eibar valencia



Figure A.2: Valencia vs Sd Eidbar. Source: 6.6



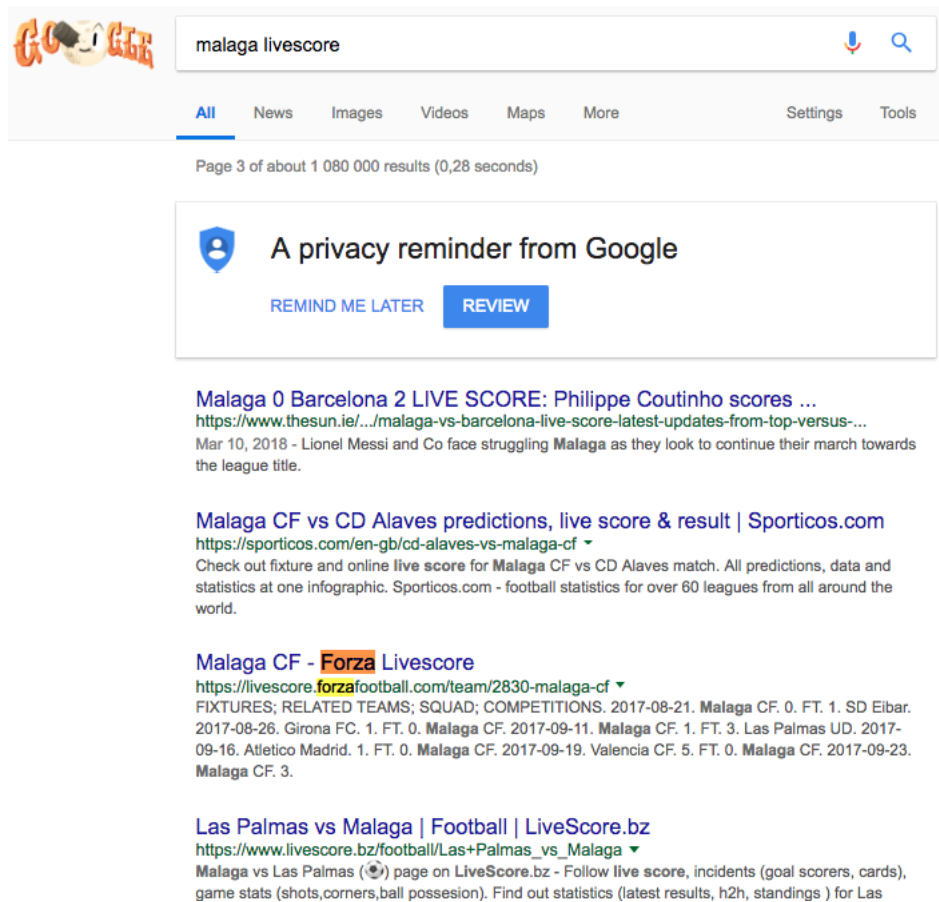


Figure A.3: Malaga livescore. Source: 6.6

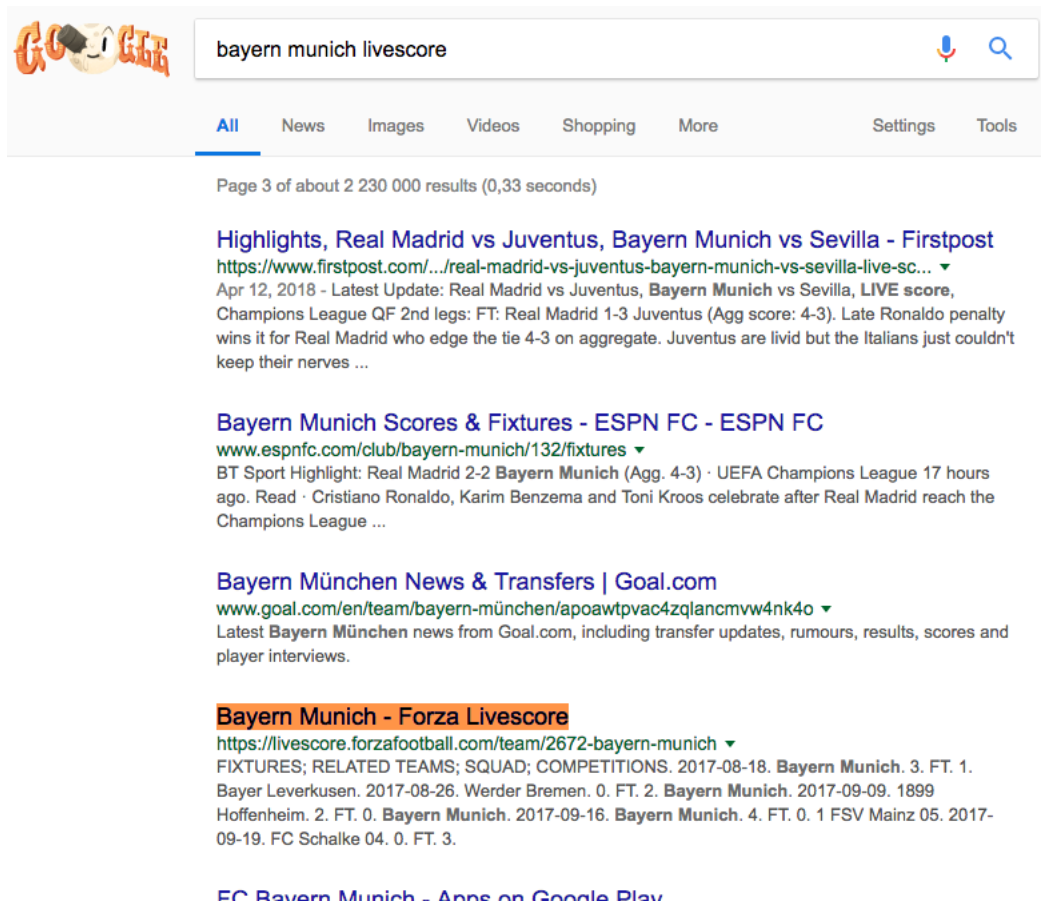


Figure A.4: Bayern Munich livescore. Source: 6.7

**SC Freiburg - Forza Livescore**

<https://livescore.forzafootball.com/team/2538-sc-freiburg> ▼

Team news and fixtures for SC Freiburg. ... FIXTURES; RELATED TEAMS; SQUAD; COMPETITIONS.  
2017-08-20. SC Freiburg. 0. FT. 0. Eintracht Frankfurt. 2017-08-27. RB Leipzig. 4. FT. 1. SC Freiburg.  
2017-09-09 ... 2017-12-10. 1 FC Cologne. 3. FT. 4. SC Freiburg. 2017-12-12. SC Freiburg. 1. FT. 0.  
Monchengladbach ...

**FC Köln vs Freiburg result livescore, 10 dec 2017**

[www.livescore.my/germany-1-bundesliga/1135867-fc-cologne-freiburg.html](http://www.livescore.my/germany-1-bundesliga/1135867-fc-cologne-freiburg.html) ▼

Dec 10, 2017 - Live score for game: FC Köln vs Freiburg (1. Bundesliga). Live result for this game, lineups, actual table and statistics.

**SC Freiburg vs 1 FC Cologne - Bundesliga - predictions, live stream ...**

<https://sporticos.com/en-gb/1-fc-cologne-vs-sc-freiburg> ▼

3 days ago - Check out fixture and online live score for SC Freiburg vs 1 FC Cologne match. All predictions, data and statistics at one infographic. Sporticos.com - football statistics for over 60 leagues from all around the world.

Missing: livescore

**Cologne vs SC Freiburg Match Preview, Betting Tips & Prediction**

<https://www.virtualbet24.com/en/Match-Preview/.../Cologne-vs-SC-Freiburg.html> ▼

Apr 22, 2018 - The match preview to the soccer match Cologne vs SC Freiburg compares both teams and includes the latest matches of the teams, the match facts, head to head (h2h), over/under statistics, table standings, match strengths and at least a computer calculated match prediction. These facts should all be ...

**Searches related to freiburg cologne livescore**

cologne - freiburg prediction	koln vs freiburg highlights
fc köln vs freiburg prediction	livescore websites
cologne vs freiburg h2h	hannover - hoffenheim h2h
fc cologne vs freiburg	cologne vs freiburg prediction



Figure A.5: Freiburg Cologne livescore. Source: 6.7

[LiveScore : Eintracht Frankfurt vs Hamburger SV](#)  
[www.livescore.com/soccer/...frankfurt...hamburger.../1-254420...](#) ▼ Översätt den här sidan  
LiveScore.com : Eintracht Frankfurt [3 - 0] Hamburger SV.

[LiveScore : Eintracht Frankfurt vs Hamburger SV](#)  
[www.livescore.com/soccer/...frankfurt...hamburger.../1-200228...](#) ▼ Översätt den här sidan  
LiveScore.com : Eintracht Frankfurt [0 - 0] Hamburger SV.

[Eintracht Frankfurt vs Hamburg | Football | LiveScore.bz](#)  
[https://www.livescore.bz/.../Eintracht+Frankfurt\\_vs\\_Hamburg](#) ▼ Översätt den här sidan  
Hamburg vs Eintracht Frankfurt (🏆) page on LiveScore.bz - Follow live score, incidents (goal scorers, cards), game stats (shots, corners, ball possession). Find out ...

[Hamburger SV soccer team - Livescore - Football results and fixtures.](#)  
[www.livescore.cz/team.php?teamid=9790](#) ▼ Översätt den här sidan  
Livescore - Place where you can find live soccer results. All is real ... Hamburger SV results and fixtures ... 05th May 18, Eintracht Frankfurt : 3:0 · Hamburger SV.

[Eintr. Frankfurt vs Hamburger LiveScore, Events & Statistics](#)  
[m.live3s.com/livescore/eintr-frankfurt-vs-hamburger-match113...](#) ▼ Översätt den här sidan  
Eintr. Frankfurt vs Hamburger Live Scores & Results, Events, Line-Ups, H2H (Head to Head), Recent Form & Statistics - Germany Bundesliga 05 May 2018.

[Eintracht Frankfurt - Forza Livescore](#)  
[https://livescore.forzafootball.com/team/2674-eintracht-frankfurt](#) ▼ Översätt den här sidan  
Team news and fixtures for Eintracht Frankfurt. ... Eintracht Frankfurt. FIXTURES; RELATED TEAMS; SQUAD ... 2017-12-12. Hamburger SV. 1. FT. 2. Eintracht ...

[Germany - Bundesliga - Standings, results, match schedule, soccer ...](#)  
[terrikon.com/soccer/germany/championship/](#) ▼ Översätt den här sidan  
Livescore ... Eintracht Frankfurt, 33, 14, 7, 12, 45, -, 44, 49 ... Eintracht Frankfurt, 3:0, Hamburger, 05.05.18 ... Schalke 04, -:-, Eintracht Frankfurt, 12.05.18 15:30.



Figure A.6: Frankfurt Hamburger SV Livescore. Source: 6.7

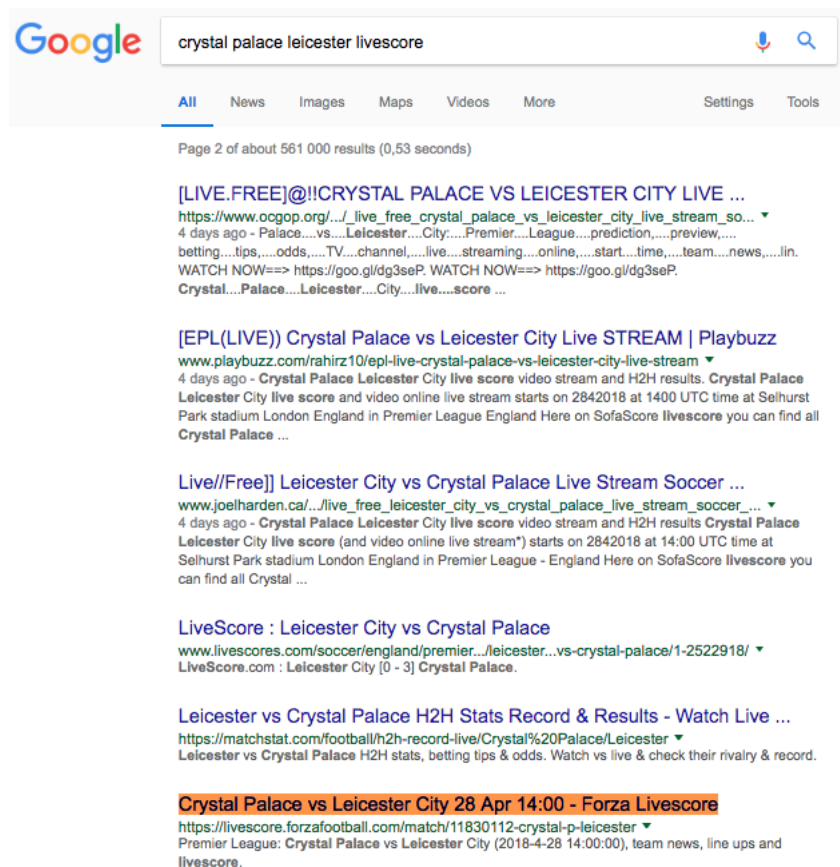


Figure A.7: Crystal Palace Leicester livescore. Source: 6.8

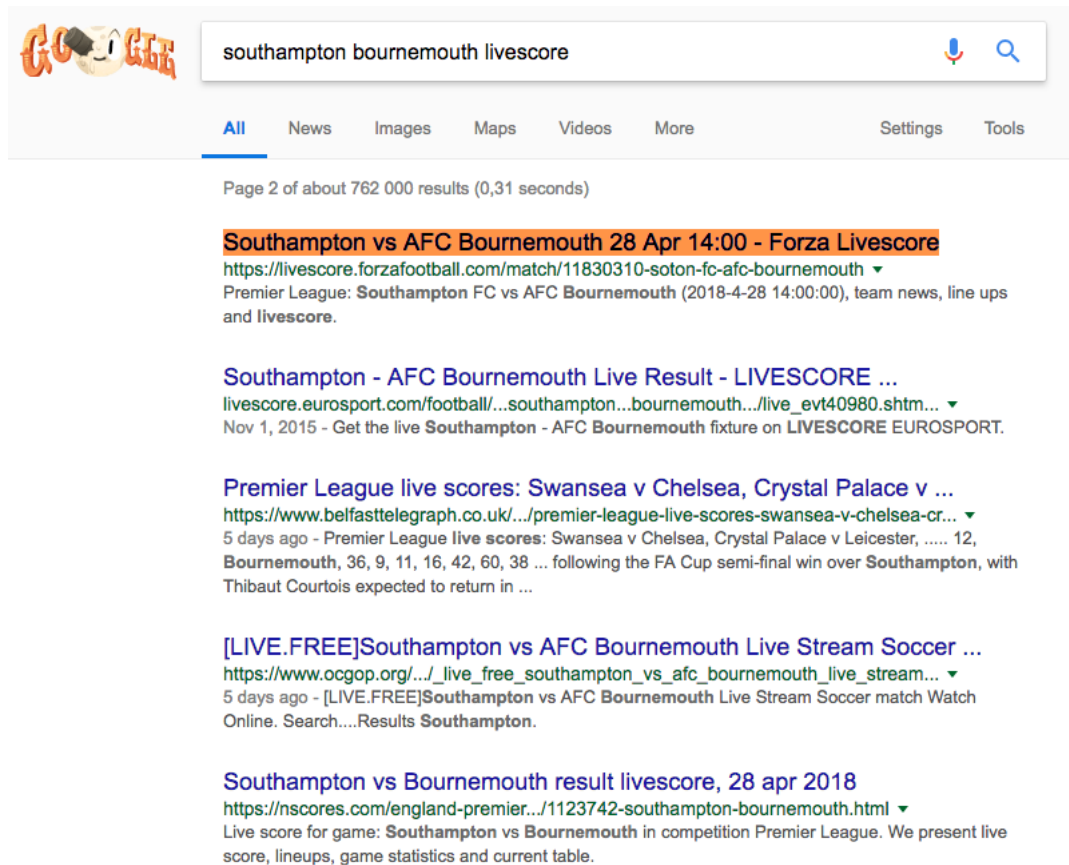


Figure A.8: Southampton Bournemouth livescore. Source: 6.8

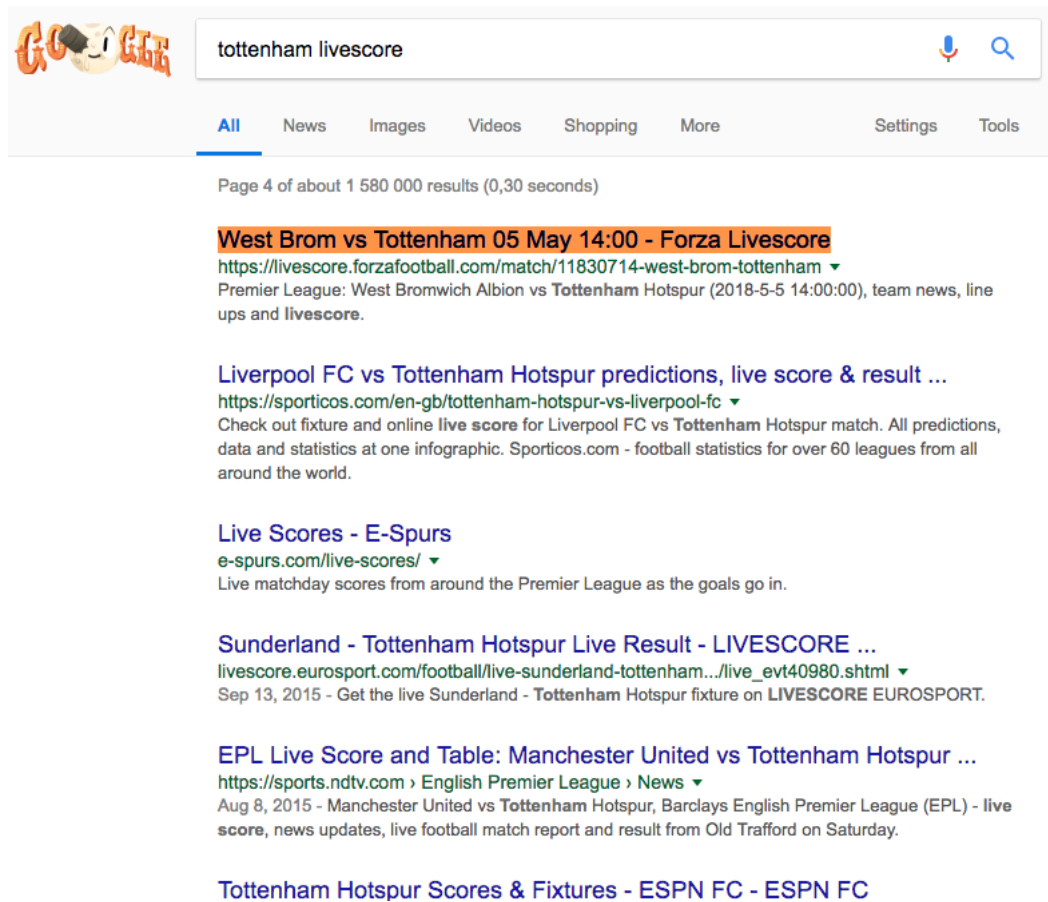


Figure A.9: Tottenham livescore. Source: 6.8