UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

LoLChrono: League of Legends Player Chronological Data

Pedro Manuel Petinga de Almeida

Mestrado em Engenharia Informática

Dissertação orientada por:
Profª. Doutora Ana Paula Pereira Afonso
Profª. Doutora Maria Beatriz Duarte Pereira do Carmo

2024

Acknowledgments

First, it is vital that I thank the Professors Drª Ana Paula Afonso and Drª Maria Beatriz Carmo, for everything they have done to help and guide me through the period of development of the Thesis. Their support and comprehension towards my situation was vital through every step of the way. Then to my parents and brother I would like to thank for all the patience given, and all the support presented when it came to helping me surpass the multiple steps of research. To my friends and colleagues who made sure I took time off when it was needed, and offered guidance and help when it was vital for me to accept it. I am in debt and will always feel thankful for having them near me. Finally, to the teams from Hospitais Universitários de Coimbra, both the Cardiology team and the Cardio-thoracic team, I am thankful for everything they did to help my health and make sure I was in the best possible conditions to continue my research. Thank you from the bottom of my heart. To all the people mentioned, I will be forever grateful.

For the players that keep playing and the researchers that keep building support tools

Resumo

Nos últimos anos, a indústria do entretenimento tem-se focado cada vez mais na sua evolução digital e, por isso, a popularidade dos videojogos tem vindo a aumentar e, consequentemente, o lado competitivo dos mesmos também tem acompanhado este crescimento. Estas competições de videojogos, ou *esports*, tendem a criar bastante interesse das diferentes partes interessadas na indústria. Mais em concreto, têm surgido várias organizações em torno das competições de videojogos. Estas organizações fazem tudo ao seu alcance para garantir os melhores resultados possíveis das competições nas quais participam. Dessa forma, existe um grande foco no processo de análise dos jogos realizados de forma a melhorar o desempenho ou corrigir possíveis erros ou hábitos dos seus jogadores. Este processo é feito, normalmente, por analistas profissionais e pelos próprios treinadores.

O processo de análise de dados é algo que também é de interesse para os jogadores casuais. No entanto, este mesmo processo não é simples para o jogador normal devido a várias razões sendo que uma das principais é a quantidade de dados criados que necessitam de análise. Esta razão, por si só, é suficiente para dissuadir muitos dos jogadores de, sequer, tentarem analisar os seus dados. Mas, no caso dos que tomam essa decisão, a parte de análise dos dados continua a ser difícil de executar, sem conhecimentos prévios. Com o nosso projeto inicial, tínhamos o objetivo de analisar trabalho prévio e desenvolver uma aplicação de web com o conhecimento adquirido. Este conhecimento viria do estudo de teses antes desenvolvidas por colegas. Adicionalmente, seria feita uma análise com foco em técnicas de análise visual de dados, que seriam aplicadas a um novo jogo ou a um novo dataset. A partir deste objetivo desenvolvemos o nosso relatório preliminar que influenciou bastante os resultados da investigação inicial, resultados estes que são apresentados no capítulo do trabalho relacionado. Para além do relatório preliminar, fez-se também a seleção do novo tipo de jogo para o qual se iria tentar aplicar este estudo. No final, foi concluído que o tipo de jogo mais propício a um estudo deste género seriam os First Person Shooter (FPS), mais especificamente o Counter Strike: Global Offensive (CS:GO) e o Valorant.

De forma a levar a cabo o projeto seria necessário a criação de um dataset para um novo jogo de forma a que se pudesse executar o projeto de análise. Desta forma decidiu-se criar um dataset para um dos jogos selecionados previamente. Caso se verificasse a impossibilidade de finalizar a análise do dataset, ainda haveria a possibilidade de executar essa análise numa implementação futura com novos projetos. Durante a execução do dataset surgiram problemas. Para ambos os jogos foram encontrados problemas específicos. Quanto ao Counter Strike (CS), a criação do dataset estava totalmente dependente dos jogadores, o que fez com que a criação do mesmo não

fosse possível, dentro do tempo desejado, devido à necessidade de contactar jogadores. No entanto, foi encontrada uma forma de transformar um id de um jogo em dataset, de forma que se conseguissem obter todos os eventos de todas as rondas. Assim que se chegou à conclusão de que o primeiro jogo selecionado não seria utilizado para a criação do dataset, considerou-se fazer a tentativa com a nossa segunda opção. Dessa forma, exploraram-se as opções de criação de dataset com o Valorant. Infelizmente, também com esta segunda opção de jogo foram encontrados problemas. Neste caso, era necessária uma chave de Application Programming Interface (API) que não nos foi possível obter. Isto verificou-se com a companhia que desenvolve e distribui o jogo. Foram feitas tentativas de contacto em ambos os casos para resolver os problemas encontrados que acabaram com respostas sem soluções possíveis. No primeiro caso, a companhia que mantinha o Counter Strike não disponibilizava os dados dos jogadores, sem ser através da sua API, e o problema acima mencionado não tinha forma de ser contornado. No caso do Valorant, foi feito um pedido para obter a chave de API necessária, o qual foi negado devido ao facto de não cumprirmos os parâmetros mínimos impostos pela Riot. Especificamente, era-nos imposta a necessidade de entregarmos um protótipo funcional para que a Riot pudesse testar e possivelmente aceitar a nossa aplicação. Devido aos problemas encontrados com os dois jogos que selecionamos para criação do dataset, não nos foi possível continuar. No entanto, todo o processo que foi desenvolvido para ambas as tentativas de criação do dataset é explorado num capítulo deste relatório.

Desta forma voltamos aos projetos feitos no passado, e ao jogo que os colegas analisaram, o League of Legends (LOL). No caso destes projetos anteriores, o foco primário eram os dados dos jogadores ao longo de todos os jogos que estes jogaram. Dessa forma verificamos que existia a possibilidade de suplementar esses projetos ao focarmo-nos nos dados temporais dos jogadores em si, e não dos seus jogos. Tendo isto em conta, o nosso foco principal passou a ser a criação de uma aplicação de web que apresentasse visualizações com os dados temporais de jogadores, ao longo de um período de tempo, o LoLChrono. Este projeto iria por si possibilitar a interpretação destes dados, pelos próprios jogadores ou equipas de pesquisa, facilitando a sua interpretação, através das visualizações a criar. Para além da aplicação, também nos propusemos explorar e explicar uma parte importante do League of Legends, o sistema de *ranked*, e fazer o mesmo para a API da Riot.

Neste relatório apresentamos informação relativa às duas empresas que estudamos e os recursos que estas disponibilizam para interagirmos com os dados dos seus jogos e aplicações. Estas duas empresas são a Valve e a Riot. Para a Riot apresentamos os dois jogos explorados, o League of Legends e o Valorant, e as duas APIs utilizadas no nosso projeto a Riot API, a API principal de onde recolhemos grande parte dos dados dos jogadores, e a DataDragon, a API onde a Riot disponibiliza todos os recursos estáticos relativos ao League of Legends. Para a Valve, apresentamos o jogo que explorámos, o Counter Strike: Global Offensive, e os dois recursos principais utilizados para recolhermos os dados dos jogadores disponibilizados pela Valve, a Steam Web API e a Valve Developer Community. Em ambos os casos todos os recursos explorados e explicados foram vitais.

Continuamos o relatório com o trabalho relacionado com o nosso projeto. Desta forma, começamos por explorar o conceito de ciência de dados aplicada a jogos e o processo de descoberta de conhecimento. Em seguida continuamos com o tópico de dados telemétricos em videojogos, onde apresentamos os vários tipos de dados que são possíveis de extrair de um videojogo. Terminamos a explorar as várias formas de visualização de dados de jogos, onde exploramos vários tipos de visualização e vários projetos que utilizam essas mesmas técnicas.

Após a exposição do trabalho relacionado, apresentamos os resultados obtidos. Dessa forma iniciamos por expor os resultados do trabalho feito para a criação do dataset de um FPS. Culminou na criação de um loop de obtenção de códigos de jogos para jogadores de CS:GO. Não foi possível terminar a criação do dataset de dados de jogos uma vez que este estava dependente de interação com os jogadores. E isso acabaria por adicionar uma dependência que, na altura, impossibilitava a obtenção dos dados de jogo de uma forma automática e linear.

Após o desenvolvimento do LoLChrono, realizaram-se testes com utilizadores para avaliar a sua usabilidade. Os participantes do estudo efetuaram tarefas específicas e no final responderam a um questionário SUS.

Palavras-chave: Dados temporais de jogadores, aplicações de web, esports, analise de dados de utilizadores, League of Legends

Abstract

In the past years, the entertainment industry has focused more and more on the digital evolution of entertainment and through this the popularity of videogames has seen a rise. Consequently, the competitive side of videogames rose along with them. These videogames competitions, or *esports* for short, create a plethora of interest from many different stakeholders in the videogame industry. More specifically competitive videogame organizations have risen in the past years. These organizations go through several steps to guarantee the best possible results come from their matches. Therefor there is a big focus on analysing past matches to evolve throughout time with the mistakes and errors made by the players. The amount of data created from these games that requires analysis is huge and that deters many regular players from trying to convey knowledge from it, in professional teams, coaches and analysts take care of this process. In previous projects, our colleagues developed an application that focused on analysing multiple matches of League of Legends and infer helpful data for both players and their teams through multiple visualization and analysis techniques. We had the goal of analysing these previous projects and attempt to apply similar techniques to new games and datasets. To achieve this, we set out to build a dataset of games for a new game type, with the intention to replicate past studies. Unfortunately it was not possible to conclude the creation of the dataset, and therefore we looked once again back to League of Legends to create a web application that could focus on telemetry data from players of League of Legends. This prototype was built to facilitate the gathering and analysis of player telemetry data. Finally we decided to evaluate the usability of our application through the use of a testing group for which we used the System Usability Scale (SUS).

Keywords: Temporal user data, web applications, esports, user data analysis, League of Legends

Contents

List of Figures

List of Tables

Acronyms

1v1 One versus One. 5

1vM One versus Many. 5

5v5 Five versus Five. 2, 6, 10, 12

ADC Attack Damage Carry. 6

API Application Programming Interface. vi, 2, 5, 10–13, 16, 24, 25, 27–29, 32, 33, 35–37, 39, 53, 54

ARAM All Random All Middle. xvi, 6, 45, 46

BST British Summer Time. 37

CLI Command-line Interface. 34

CS Counter Strike. v, vi, 12, 13, 24, 55

CS2 Counter Strike 2. 13, 55

CS:GO Counter Strike: Global Offensive. v–vii, 2, 3, 5, 12, 13, 23–25, 53–55

Dota Defence of the Ancients. 12

DPS Damage Per Second. xv, 18, 19

DS2 Dead Space 2. 21

EA Electronic Arts. 21

EUW Europe West. xix, 8, 9, 47

FPS First Person Shooter. v, vii, 2, 3, 6, 10, 12, 23, 24, 53

GUI Graphical User Interface. 36

GW2 Guild Wars 2. xv, 18, 19

HTTP Hypertext Transfer Protocol. 35, 36

IP Intellectual Property. 6

JSON JavaScript Object Notation. 25, 36, 37, 45

KDA Kills Deaths Assists. 45

KPI Key Performance Indicator. 17

LOL League of Legends. vi, ix, xv, 1–3, 6–8, 10, 11, 13, 26–28, 38, 45, 47, 48, 54

LOR Legends of Runetera. 6, 10, 54

LP League Points. 3, 7–10

MMORPG Massive Multiplayer Online Role Playing Game. 18, 54

MMR Match Making Rank. 3, 7, 8

MOBA Massive Online Battle Arena. 1, 2, 5, 6, 27, 28, 47, 48

MVC Model-View-Controller. 30

MvM Many versus Many. 5

NOSQL Not Only Structured Query Language. 36

PC Personal Computer. 28

RSO Riot Sign On. 38, 39, 54

SUS System Usability Scale. ix, xvi, 46–51

TF2 Team Fortress 2. xv, 20

TFT Team Fight Tactics. 6, 10, 54

UI User Interface. 34, 35, 54

URF Ultra Rapid Fire. xvi, 45, 46

URL Uniform Resource Locator. 25, 26

WoT World of Tanks. xv, 19, 20

Chapter 1

Introduction

This chapter serve as the introduction for the study. In Section 1.1, we will present the motivations behind the project. In Section 1.2, we will present the main objectives of the thesis. In Section 1.3, we will present the contributions for the work developed. And finally, in Section 1.4, we will go over the structure of the of this document.

1.1 Motivation

For the past years media has evolved quite rapidly, and this development is also portrayed in the video game industry. Recently video games have become a more common hobby. Along with the rise of video games, the eSports scene has been steadily growing throughout the years, with viewership peaks being surpassed year after year [28].

As games evolved, and their popularity rose, it has become more common to collect data, such as telemetry data from those same games, in order to, not only facilitate continuous development [38][64], but also, so that the players can see how they have evolved throughout their playtime. For this study, the main focus of interest, is the player base's data, since it is from this data that we will obtain the required information for the conclusions. This data is important for the growth of all players alike, whether they are a more casual player which strives to rank up and learn how to play the game in a better way, or whether they are a professional player tasked to compete while representing a team, this data, when used correctly, can make or break a match [30].

When it comes to past projects, we had the past thesis by our colleagues from the faculty, Costa [11], Afonso [1], Moucho [42], Vieira [60], and Carreiro [7] which applied multiple techniques in visualization of data and data analysis to the popular Massive Online Battle Arena (MOBA) game League of Legends (LOL). These past thesis focused mainly in the analysis of game data. This data would then be presented to players through the most appropriate methods, such as, interactive map visualizations, analysis of game events, and other methods, in order to aid players and coaches analyze and understand their behavioural patterns to enhance their performance and grow better in the game. These methods could be utilized with a singular match or with a plethora of matches, and could also have a specific player has a point of focus or utilize multiple players in a team to analyse their performance as a team. In our case, we wanted to study this approach and attempt

to replicate their success with a different game, of a different gametype, specifically with a game of the First Person Shooter (FPS) genre. This option was explored, but eventually we hit a bump that made it unfeasible. Therefore, the scope of the thesis changed and landed in the creation of a web application that could search and display playtime information for players of LOL to better understand the player's gaming behaviour. This would then facilitate the gathering of data when it came to playtime statistics in future studies. Along with the display of playtime metrics, we also focused on presenting several visualizations that could convey useful data in an appropriate way.

1.2 Objectives

The initial objective of this thesis was to study the work done by previous students of the Faculty on VisuaLeague, these students being Sonia Costa in 2022 [11], Rafael Afonso in 2020 [1], Tiago Moucho in 2018 [42], and Pedro Vieira in 2017 [60]. We would then attempt to apply the same concepts to a different game. The previous studies were done using the popular Five versus Five (5v5) Massive Online Battle Arena game League of Legends while the new game genre which we wanted to explore was the First Person Shooter games such has Counter Strike: Global Offensive (CS:GO), Valorant, or Rainbow Six Siege.

We had the following main objectives:

- Study the concepts of Visual Data Analysis, Spatio-Temporal Data, Trajectories, and Data Clustering;
- Apply the techniques used in VisuaLeague to a game of the FPS genre;
- Propose possible visualizations for the game chosen.

Mainly we focused in the gathering of event data for games in the FPS genre:

- Explore the existing services for review of matches for FPS games, and services that show statistical data for FPS games;
- Explore the Application Programming Interface (API)s provided for the games and the services where those games are hosted, in order to then create datasets for those games;
- Find and implement a workflow for data gathering for the chosen game CS:GO.

After the possibilities to gather data were explored, we reached the conclusion that it was unfeasible to gather data for FPS games. This was due to an impossibility to automatize the process. So, we once again, turned our attention back to League of Legends to develop a project with a focus on Temporal Data for players.

Therefore the final set of objectives evolved into:

- Explore the Riot API for data gathering and present an understanding of the restrictions applied by Riot for the access to their API;

- Create a web application that could be used to gather and present player playtime data gathered from LOL players;
- Propose appropriate visualization techniques to extract knowledge from the gathered data;
- Test the application with a handful of players to create a proof of concept for the application.

1.3 Contributions

The main contributions of this thesis are the following:

- A study and discussion of the previous research on Visual Data Analysis, Spatio-Temporal Data, Trajectories, and Data Clustering;
- An in-depth analysis of the Ranked system in League of Legends, which explains the Ranked Ladder, the concept of League Points (LP), Match Making Rank (MMR) both visual and hidden, the Decay System that exists in the higher ranks in LOL for the time at which the data was gathered, the first split of the season of 2023, and for the in-game ranked system, of that same season;
- An in-depth study of the methods to gather CS:GO match replays, and the workflow needed to get match data, along with the process to extract the event data from the match replay;
- The implementation of a web application, dubbed of LoLChrono, which shows temporal data of players for the split one of the season of 2023 of League of Legends, showing stats such as play time, matches played, and sessions played.

1.4 Structure of the Document

We will now go over the structure of the remaining of the thesis and give a small summary for each of the remaining chapters in this document.

In Chapter 2 we will go over the background information that we find necessary for the project. This involves a section where we go over eSports, the rising popularity they had in the last couple of years and some of the games that are played in these competitions. Following this, we talk about the two companies for which we explored games. These companies are Valve and Riot. We give a small introduction to the story of these companies and we go over some of their games and services.

In Chapter 3 we present the Related Work found for several topics that we consider important for the thesis. These topics are Game Data Science, Telemetry Data in Video Games, and Visualization of Game Data. We consider these to be the vital topics to go over so that we can more easily follow up with the presentation of the project.

In Chapter 4 we present the work done, when it comes to the attempt at building a dataset for an FPS game. We then go over the games for which we tried to build the dataset for, and we also

discuss some tools which either already do some sort of analysis, or enable data gathering for the games. We finish the chapter by presenting the results and conclusions for this project.

In Chapter 5 we present the analysis and development of the LoLChrono application. This includes the description of the prototype, the initial and final architecture, the technologies used that enabled the development of the prototype, the methods used to gather and keep data, we then present the design choices of the project, and we conclude by presenting the user evaluation of the prototype.

In Chapter 6 we present the conclusions and future work for the project. These are vital for the viability and future of the project.

Chapter 2

Background

In this chapter we will introduce some background concepts that are approached throughout the thesis which are important for our project. In Section 2.1 we will go over eSports and some of the popular games played in those eSports competitions, along with an overview of the rise of popularity from both video games and their eSports counterpart. In Section 2.2 we will talk about Riot Games, and the games and API services that where explored during the duration of the project. Finally in Section 2.3 we will talk about Valve Corporation, Counter Strike: Global Offensive, and the APIs that Valve maintains to communicate with their platforms.

2.1 eSports and their Games

The rising popularity of video games, and with them the competitive scene that they bring along with them known as eSports, has brought them into the spotlight in the past years as an interesting object of study [44][5][28]. In the past years, Massive Online Battle Arena, Shooters, Battle Royales and many other video game types have seen a rise of player bases, due to their competitive nature [8]. ESports are a form of competitive tournaments where teams or singular players compete against one another, in one of several formats of One versus One (1v1), One versus Many (1vM), or Many versus Many (MvM) format where the last man, or team, standing wins. These formats are highly dependent on the game played since each game, typically, has their own set of rules. With this evolution on both the games and eSports ends, the ability to collect and gather data from the matches played also had to keep up with the rise of interest. Through the use of data collection techniques, it is possible to acquire multiple datatypes from these games, such as, spatio-temporal data, statistical data, and event data. This data can then be utilized to analyze player performance which is a crucial part of developing talent in both the competitive and casual environments. In a professional context, teams try to enhance their players' synergy, in order to counteract the teams they go up against. While in a casual context, players look for ways to rank up the ladder in order to reach their peak rank and attempt to climb past it. Through the use of every resource, in both cases, they try to find and understand their flaws and mistakes [30].

2.2 Riot Games

Riot Games was founded in 2006, with the main objective to develop, publish, and support the most player focused games in the world. In 2009 their debut game was released League of Legends, in the following years LOL would rise to become the most popular played MOBA [22]. Due to this success, Riot Games was able to spread their range of offers, not only in the game sector, but also by forging a massive Intellectual Property (IP) which gave birth to new titles such as Valorant, their take on a First Person Shooter, Legends of Runetera (LOR)their take on a card game, and Team Fight Tactics (TFT) their take on an auto-chess game, all these titles continued to expand their world but, this was not achieved purely through the games they developed. At the same time these games were created and rose in popularity Riot continued to build an amazing world through their story telling and multiple multimedia projects across music videos, comic books, board games, and their extremely popular animated series Arcane. Besides all this, we also cannot forget their eSports scene which culminates every year in the League of Legends World Championship which is seen as one of the biggest eSports events worldwide [22].

2.2.1 League of Legends

League of Legends is the biggest and most active game of its genre. League of Legends is a Massive Online Battle Arena type game, or MOBA for short. The main objective in LOL is to destroy the opposing teams nexus. In order to achieve this, each team picks 5 different champions between themselves, and engage in a battle against the opposing team for resources that will upgrade the champions abilities and items that will enhance their power. There are several ways to play LOL, the traditional 5v5 Summoner's Rift, All Random All Middle (ARAM) a game mode with all random champions where there is only one Lane to fight for, Training Modes such as Bot games or the Training Tool, and finally the Event game modes, of which there are many and all with different rule sets. These are important to mention has the data output from these is different, but also gather-able.

Out of all the game modes, 5v5 Summoner's Rift is the most played game mode in LOL, we can see the map in Figure 2.1, but there are several ways to play it. Once you choose Summoner's Rift, you can also decide between playing the casual way of, Blind Pick and Draft Pick, or the competitive way, where rank comes into play with Ranked Solo/Duo and Ranked Flex. The base rules of the game do not change between game modes, what does change, is the pick order and the existence of ban phases in Draft Pick but not in Blind Pick, and between the competitive game modes what changes is the party sizes and the different ranked ladders. In Solo/Duo Queue a player can play the game by themselves or accompanied with a friend, while in Flex Queue there is also the possibility to play has a team of three or a full team of five.

We should also talk about the game and the different roles. In a standard match of 5v5 Summoner's Rift, there are five roles which are linked to the four lanes. The roles are the following Top laner, the Jungler, the Middle laner, and then we have the Bottom lane duo of the Attack Damage Carry (ADC) and Support. These roles differentiate in terms of the lanes the players go to, while

the champions they play can differ along with the team's strategy.

Figure 2.1: Image with the main game mode map of League of Legends Summoner's Rift

Ranked System

When we talk about Ranked LOL, there are three statistics that we must have in mind, these are Match Making Rank (MMR), Rank and League Points (LP). These are the stats that determine the skill at which the player is playing at, although while MMR is a global hidden stat that every account has but none can check, LP are linked to the ranked system and along with the Tier and Rank, represent the spot at which the account is on the ranked ladder. Before we go into the depths of the ranked ladder, we must understand the Tiers and Ranks of that same ladder, and along with those the definitions of MMR, Rank and LP.

So the ranks in League of Legends, for the Season of 2023 split 1, have the distribution that can be seen in Table 2.1.

Table 2.1: Tier and Rank disparity along with rules for Promotions and Demotions for the Season of 2023 Split 1 at the beginning of the season

Tiers	Ranks	Promotion Rules	Demotion Rules
Iron	From I to IV	100LP per rank - Promotion games	Fall under 0LP
Bronze	From I to IV	100LP per rank - Promotion games	Fall under 0LP
Silver	From I to IV	100LP per rank - Promotion games	Fall under 0LP
Gold	From I to IV	100LP per rank - Promotion games	Fall under 0LP
Platinum	From I to IV	100LP per rank - Promotion games	Fall under 0LP
Diamond	From I to IV	100LP per rank - Promotion games	Fall under 0LP or Decay
Master	0 - 200 LP	More LP than lowest Grandmaster	Fall under 0LP or Decay
Grandmaster	200 - 500 LP	More LP than lowest Challenger	Fall under 200LP or Decay
Challenger	Infinite LP	Infinite LP	Fall under 500LP or Decay

Now that you have seen a simplified look of the ranked ladder, there are necessary concepts to understand the ranked system in a more thorough way.

First let's talk about what MMR is. Every player of League of Legends has an MMR stat linked to their account, but this stat is only known to League of Legends. There is no way to access your MMR, one can have a good guess about where it hovers but never what it is precisely. But what is MMR? Well MMR is what Riot uses to determine the players place in the ladder, and it is the stat used for matching players of similar skill levels [33]. So, we can describe MMR as the skill level for a specific player attributed to them by League of Legends.

Now let's talk about Rank. Rank is the skill level at which the player is. In other words, to obtain a Rank a player must play Ranked Matches, depending on how well he does his Rank can rise or fall. This stat is linked very heavily to the LP stat, since one can only climb the Rank ladder if he wins LP and fall if he loses LP. Losing and gaining LP is synced to winning or losing Ranked Matches [33]. There is also a very strong link between Rank and MMR, since the visual Rank should end up being the same has the player's MMR it is easier to climb when you are ranked under your MMR, and harder when you surpass it, since the game sees you in a higher skill level than you should be placed at [33].

Finally let's talk about LP. LP is the number of Points one has in a specific Rank, depending on the rank the player is at, these points will fluctuate between 0 and 100. For high ranks, the rules are different with a linear climb of LP up to however much the player can get to [33].

Promotions and Demotions are also something to consider when we talk about the ranked system. In low ranks, Diamond and under, the rules for promotions and demotions are very standard. If the player gets 100 LP in their rank, he qualifies for a rank or tier promotion. He then enters the promotional matches of which he has to win a best of three queue to rank up, if he loses the best of three he maintains his rank and can try again once he reaches 100 LP. But there is a small detail to note, if the player reaches 100 LP consistently, but always fails to rank up, he will be awarded a free win in his promotional best of three. Demoting is not much different from ranking up, once a player hits 0 LP he is in danger to get demoted. Depending on how his MMR is, the demotion will be quicker or slower. What this means is that a player might not get demoted instantly once he loses a match at 0 LP [34].

Promotions and demotions in the Apex Tiers of the Ranked ladder, Master, Grandmaster and Challenger, work differently. For once these players have new limitations introduced to them in terms of queueing, for solo/duo queue they must queue alone, and in flex queue they can queue in groups of one, two, three or five but cannot queue with players ranked under Platinum. Now when it comes to promotion and demotion rules in high tier, they are more fluid. At this point it is only a grind for LP, and once the clock hits zero, in local time, the ranks are recalculated and attributed to the players.

Although there is a limit on the number of players that can hit Grandmaster and Challenger tier, these limits depend on the server the player is playing on, but as an example we can see the Europe West (EUW) region player limitation in Table 2.2, and the remaining servers will presented

in the appendixes [50].

Table 2.2: Europe West (EUW) Region player limitations for Challenger and Grandmaster Ranks [50]

	Solo/Duo Queue	**Flex Queue**
EUW	Grandmaster - 700 Players	Grandmaster - 500 Players
	Challenger - 300 Players	Challenger - 200 Players

It is also noteworthy that high rank players must also keep attention to Decays and their Banked Days. These are two mechanisms that are introduced in the Diamond rank and exist only for that rank and those above it. But what is Decay, and what are Banked Days? Well let's go through these new mechanics.

Decay is a mechanic that is introduced in higher ranks to verify that active players are rewarded by playing the game and inactive players suffer the consequences of their inactivity. Decay is felt differently depending on the rank you have, although what it does is quite linear. If a player does not play a game per day in the Apex ranks, then they suffer Decays, which means they lose a set amount of LP. This can be avoided with Banked Days [34].

Banked Days are acquired by playing multiple games in a singular day. Depending on the rank of the player the rules for obtaining Banked Days change [34]. The rules for Decay and Banked Days are the following.

In Diamond Tier we have the following rules:

- Banked Days per Match: 7;
- Maximum Banked Days: 28;
- Initial Days Before Decay: 28;
- LP Lost on Decay: 50.

While in the Apex tiers of ranks, the rules are:

- Banked Days per Match: 1;
- Maximum Banked Days: 14;
- Initial Days Before Decay: 14;
- LP Lost on Decay: 75.

Note, that if a player falls to or under 0 LP due to decay, they will be placed in the next lowest division of rank. Although you may also rank down into the lower tier of ranks (i.e., from Diamond to Platinum). And, if an Apex tier player decays from inactivity, they will be placed into Diamond II, and will have to restart their climb on the ranked ladder from there.

Finally, we must keep in mind that the ranked system is an ever-changing system. This is visible if we check the changes done for the current season (Season 2023 Split 2), where an

additional rank tier was added between Platinum and Diamond, called Emerald. And additionally, the Promotion game system was removed and tiering/ranking up has become a more fluent process, where once a player gets to 100 LP, he will rank up immediately. Also besides the LP rank changes, the placement games needed to get a rank have been halved, previously 10 was changed to 5 [48].

Throughout this subchapter the description of the Ranked System for the Season of 2023 Split 1 of League of Legends was presented but, we must keep in mind that the Ranked System of any game will always be an ever-changing system, therefore what is described here might not be true in the future. In fact, Riot has already shown intention to continue their work refining the Ranked System they have whether it is through the change of the dates for the multiple splits, or by adding and changing the multiple mechanics that go into the climbing process [45].

2.2.2 Valorant

Valorant is Riot Games' take on the First Person Shooter genre. It combines traditional FPS elements with an Agents system to create a First Person Tactical Hero Shooter. Originally teased as Project A in October 2019, Valorant was officially released on June 2 2020 [23][24][25].

Valorant is a first to 13 rounds style Five versus Five hero shooter, where both gun play and ability usage defines the outcome of a round. Agents, the heroes of this game, battle in a 5v5 round base format with a single life per round, where the objective of the attackers is to plant a spike and have it detonate, while the defenders do their best to avoid it to happen, both the planting of the spike and it's eventual explosion. When it comes to Agents, each has their own unique set of abilities, but they all fit within four genres of Agents. Duelists are best suited to attack opponents. Initiators are best suited to begin engagements. Sentinels are best suited to hold down site or cover flanks. And Controllers are best suited to take control of parts of the map. When it comes to weapons there is a buy phase at the start of each round where every player can buy their arsenal. And finally we must talk about the maps, when it comes to maps every map is unique and therefore adaptation is the key to victory [31].

2.2.3 Riot API and Data Dragon

Through the use of the APIs maintained by Riot, the Riot API and Data Dragon, we have access to all the necessary information to build our applications and datasets. We can learn how to access this information through the Riot Developer Portal [17], which is the best place to visit for information on how to build third party applications, that want to use Riot information. Before we analyze each API, we should bear in mind that the Riot API is used for information on any of the Riot games. Whether that is League of Legends, Valorant, Legends of Runetera, or Team Fight Tactics, when it comes to data on the games the Riot API is where we need to search. While Data Dragon is a secondary web API that maintains patch information on League of Legends alone, which can be used for static resources.

When it comes to the Riot API [18], there are restrictions to access certain data. In order to

control the access to their data, Riot implemented an API key system that should be understood. There are three types of API Keys, there are Development Keys, Personal keys, and Production keys. All of these have different restrictions set on them, for acquiring the API key, for limiting the rates at which requests can be done, or what needs to be done in order to maintain the API key [19]. So let's go over each of the keys.

First we have the Development keys, these can be generated by every Riot account, and enable the users to tinker with the Riot API and do research for personal projects. The major issues with these keys is their lifetime, the keys, once generated, have a lifetime of 24 hours and then become deprecated. When it comes to rate limitations, they have the same limitations has Personal keys [19].

We then have Personal keys, these keys require for the user to register them to a product. Once the key is linked to a project it can only be used for that project and the project can't have any other key. The product needs to be registered through the Riot Developer Portal. When it comes to acceptable products to register for a key there are also some restrictions, Riot presents some possible products that are allowed to register for keys [19]:

- Bots for streaming sites, boards, voice com servers, etc ...;
- To display your own personal stats for your personal website;
- Personal projects to gather your own stats;
- Personal research;
- Projects meant for personal usage and not production.

There are also limitations when it comes to request rates, for both Development keys and Personal keys, the rates are as such:

- 20 requests every 1 second;
- 100 requests every 2 minutes.

Finally we have the Production keys, these require the users to register the product and wait for the approval of Riot to acquire the key. For this, the product has to be hosted in order for the employees to test and approve the products functionalities. The reward for upgrading to a Production key is the higher limits for the requests, which become has such [19]:

- 500 requests every 10 second;
- 30000 requests every 10 minutes.

Besides the Riot API we also have access to Data Dragon, which is Riot's response to requests for LOL patch data and game assets. This API has static data with information for several data types such as, champions, items, runes, summoner spells, and profile icons, so when it comes to data that isn't updated regularly it is all centralized on Data Dragon. There are a couple of details

that should be considered, the information on Data Dragon is manually kept which means that every patch it needs to be changed and the time period it takes to update may vary. Besides this we also have to take into consideration that Data Dragon changes depending on the region of data requested [20].

2.3 Valve Corporation

Valve Corporation was founded in 1996 by two ex-Microsoft employees Gabe Newel and Mike Harrington. Valve started its life with a focus on game development, releasing titles that influenced or created genres of games. Three of the more popular game franchises by Valve are Half Life, Counter Strike (CS) and Defence of the Ancients (Dota). Now a days, Valve continues their focus on game development, but have also turned their attention towards their most popular platform Steam and also actively take part in hardware development with several popular products in the market such as the Steam Deck, a handheld platform to play Steam games and others [55][35][65].

Besides the products developed by Valve, whether they are physical or digital, they also have responsibility over a couple of resources intended for third party developers to interact with their products, and even facilitate the creation of new products using Valve technologies. In our case we mostly looked at technologies that helped us gather data from CS:GO matches. With this intention we explored the several platforms that Valve created to acquire data from the CS:GO matches. But before we talk about the process found, we should go over the platforms that exist and what their major focus is. The Steam Web API [56], is the platform that is kept for interaction with Steam. Using this API we can get information on the several parts of Steam. We can obtain information on the games that are hosted and sold on Steam, the players and their account information, or the Marketplace and the News section for games. There is also the Valve Developer Community [10] which is a wiki like page where multiple developer tools and guides are presented to the developers that visit the web-page. Through the information found in this page we where able to acquire CS:GO game codes. Finally, they have SteamWorks [57], which is a set of tools and services built by Valve that help configure, manage, and operate games on Steam.

2.3.1 Counter Strike: Global Offensive

CS:GO is a multiplayer tactical, First Person Shooter competitive online game developed by Valve Corporations [55] and published in 2012, which mixes the traditional shooter mechanics with an objective based 5v5 round style gameplay. It is also noteworthy that it is the fourth iteration in the Counter Strike series of games. It normally involves two teams of five players, Terrorists and Counter terrorist, in a series of objective-based gamemodes. The most common gamemode is a Search and Destroy style game, where the Terrorist team attempt to plant a bomb and the Counter Terrorist team is tasked to stall its opponents or defuse the bomb. The game offers multiple gamemodes and maps, although the most common mode played is competitive which is the Search and Destroy style mode. When we consider maps, it is complicated to present a list of the most played maps, but you can consider that the total map pool for the competitive mode is ten, these are

Inferno, Mirage, Nuke, Overpass, Vertigo, Ancient, Anubis, Dust II, Train, and Cache. The players can freely handpick the maps they want to play on or can select to play in a professional match environment where a map pick and ban phase occur. These maps all have different layouts, and that means that not all get played the same. It is also important to mention, that has of September 27th 2023 CS:GO is no longer available due to the update to Counter Strike 2 (CS2) the new CS title.

2.3.2 Steam Web API and the Valve Developer Community

Two of the platforms mentioned in the introduction of Section 2.3 were vital for the data gathering process that was attempted for CS:GO. These are the Steam Web API [56] and the Valve Developer Community [10]. With the articles found in the Valve Developer Community we where able to implement a small script that interacted with the Steam Web API to obtain information on CS:GO players and the matches they played. This process will be more thoroughly described in the Section 4.3 since it was a lengthy process which integrated several different technologies besides the APIs mentioned. These two APIs are nevertheless the main parts of the process and are vital for any kind of prototype that wants to interact with Steam information or Valve games.

2.4 Summary

Throughout this chapter we have explained the importance of eSports for the evolution and development of videogames overall. Not only this we also discussed both of the companies we explored in this study. For Riot we analyzed League of Legends and the ranked system for the game, along with a small overview of Valorant and the APIs maintained by Riot itself. Following this, we also talked briefly about Valve, their history of products and their services. And we concluded by analyzing Counter Strike: Global Offensive and the APIs maintained by Valve.

Chapter 3

Related Work

In this chapter, we will go over the past research done on several topics that are relevant to the thesis. In Section 3.1, we will go over Analysis of Game Data for both video games and sports. In Section 3.2, we will have an insight into telemetry data when it is applied to Video Games. Following that in Section 3.3, we will go over the techniques used for Visualization of Game Data and the most popular methods to present data whether it is trajectory data or telemetry data.

3.1 Game Data Science

From the games played, both through eSports competitions and casual play, there is a plethora of data that is generated [30]. From all this generated data a lot of intelligence can be gathered, which can be of use for multiple actors, such as game developers and the game players. The use of telemetry data for the actors will be seen in a future section (Section 3.3), but before, we will go over the process of game data science. In order to derive knowledge from data we can adopt the scientific method, which in data science has been adapted to the analysis of data via framework which is referred to as the Knowledge Discovery Process [12]. Within this process there are two ways to infer reasoning, this can be done either through inductive or deductive reasoning [12]. Deductive reasoning is a top-down approach where a conclusion is obtained from a set of rules or a theory. It is described as a narrower and more of a test concerned, or as a hypothesis confirming approach which normally ends up being more useful at the end of the discovery process as a confirmatory analysis tool. Meanwhile, inductive reasoning, is a bottom-up approach where conclusions are reached from specific observation, it is described as more open-ended and explanatory, which is more useful at the beginning of the discovery process [12].

The Knowledge Discovery Process is described by Seif El-Nasr et.al. [12] as a circular process with several methods, which are Attribute Definition, Data Acquisition, Data Processing, Metrics Development, Analysis and Evaluation, Visualization, Reporting, and Knowledge Deployment, as can be seen in Figure 3.1, each of these methods have their own importance and all are required for the process to be completed [12].

Attribute definition is the initial step where the variables and tracking strategies have to be set, in our case we set the attribute of focus as time focused variables such as time played. Data

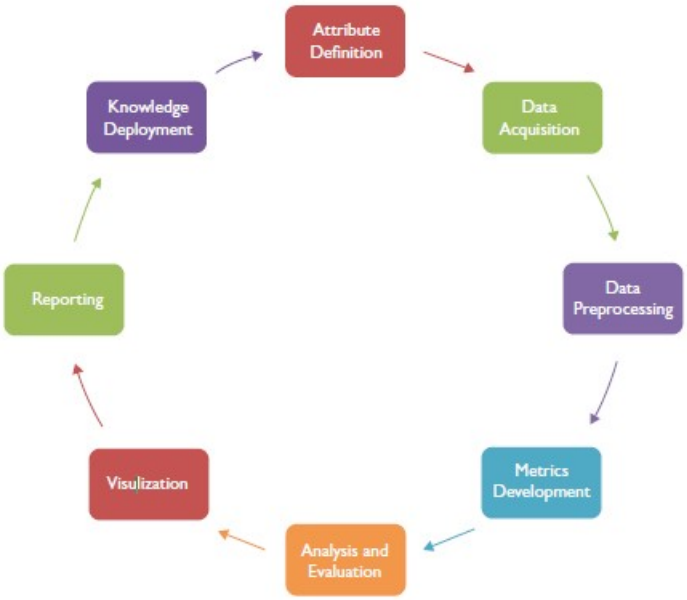


Figure 3.1: Applied process of Knowledge Discovery by Seif El-Nasr et.al. in [12]

acquisition is the process where we gather data to analyse, in our case we did the research required to reach the APIs used and the calls that where useful in said APIs (Section 5.4). Data processing is the process where we analyse the data gathered and identify inconsistencies and incoherence to then fix these in order for the data to be usable. Metrics development is the process where meaning is constructed from the data or the abstractions are built. In our case we looked at time series data and tried to formulate hypothesis based on our proof of concept (Section 5.5.5). Analysis and evaluation is the process were the information gathered is analysed and evaluated in order to either accept, reject or refactor the hypothesis constructed. Visualization is the process where we have to consider the best ways to visualize the data gathered, this is a vital step since without proper visualization data can be lost. Finally we have reporting and knowledge deployment, which, are joined together due to both being connected to the process of presenting the discovered knowledge to the relevant stakeholders, in our case this report is our way to deploy our knowledge but more specifically we present the knowledge acquired in Chapter 4 and Chapter 5 [12].

3.2 Telemetry Data in Video Games

Telemetry data is all the data that we can acquire from a distance. Therefore, in the context of video games, it is all the data that can be gathered by the game developers and then distributed through APIs. When it comes to video games, the types of data that the companies decide to provide, such as event data, spatio-temporal data, statistical data and any other data type that is of interest, can be considered telemetry data. This data is of interest to game developers, and other actors [12]. With this said, depending on which actor looks at the telemetry data, many different types of conclusions and information can be withdrawn from this data. Developers, can use telemetry data to balance their games or learn from past projects to enhance and upgrade their future projects.

Researchers, can use telemetry data to study player behaviour or for the study of the development and update cycles that companies employ, and with this understand how companies upkeep their live services and video games. Players, can also have interest in this data, since they can obtain knowledge from interacting with their data. Not only to learn from their past match data but also to try and rank up by learning from their past mistakes or their triumphs in order to enhance their teamwork, and personal skill [38][30][61][13][64][36].

So when it comes to analysis of telemetry data, it can be focused on game developing or on player data. These methods can focus on different types of data depending on their main objective. For example two different developing teams which work on the same game can look at different types of data to be able to achieve their goals. More specifically statistical data for weapons in a shooter game can be used to balance the meta in a live service type game, and spatio-temporal data can be used for the same game to investigate if the maps are being traversed as they are intended to, this can be especially useful during game development. While, if we look at the player data, having their data accessible for tracking can be a motivation towards players optimizing their gameplay, or by comparing data between friends it can encourage competition between themselves. When it comes to the methods used for presenting player data it can be done through visualization of statistical data, heatmaps, replay analyzers, or summary visualizations for replay modes [61].

Telemetry data is therefore the raw data we can acquire from videogames, this data needs to be interpreted into game metrics in order to have quantitative measures or objects from which we can then acquire knowledge from. These game metrics can also be seen has Key Performance Indicator (KPI), which are strategically selected metrics that can demonstrate how certain objectives are achieved. These KPIs can belong to three broad classes of metrics, player metrics, performance metrics, or process metrics. Player metrics are related to the people that play the game. Common types of analysis using these metrics are time spent analysis, trajectory analysis, or social network analysis. Performance metrics are related to the performance of the technical infrastructure that maintains the game. And finally process metrics are related to the process of developing games and managing the creative process of the development methods [12].

As we can see on Figure 3.2 within player metrics we have other more specialized metrics. These are community metrics, customer metrics, and gameplay metrics. Community metrics are more focused on the social dimension of players. Customer metrics are more focused on the aspects that cover a player as a customer. And finally, we have the gameplay metrics, these focus on the player and their behaviour inside the game, these are the metrics we will focus on, or at least some of the metrics within other sub-branches of gameplay metrics [12].

Within gameplay metrics, exist several sub-branches of metrics, these are the following. Engagement metrics refer to a players commitment to the given game they are playing. These are some of the most important for our project since we are mainly focusing on metrics that have to do with time played, our project explores multiple engagement metrics such as, days played, average daily playtime, distribution of playtime, evolution of average daily playtime, and session length and frequency. We then also have acquisition metrics, which focus on new players, in our case

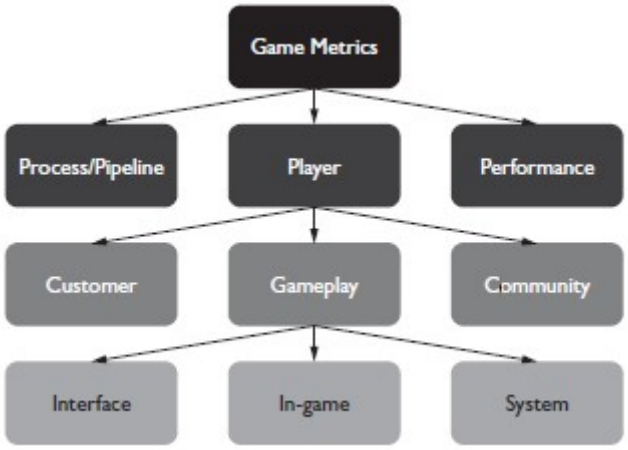


Figure 3.2: Different types of behavioral metrics for the player, performance, and process categories by Seif El-Nasr et.al. in [12]

these are not as important. Following those we have retention metrics, which can be of interest since they focus on maintaining active players, so things like the churn rate of a player can have some interest of study. Finally we have progression metrics, which are used to gauge the progress of players through the game, these can be used if we related playtime to progression [12].

3.3 Visualization of Game Data

As seen in Section 3.2 game data can have different types of focus, but the way this data is visualized will differ on the perspective used to look at the data. Instrumentation of games to automatically collect game metrics has become an important aspect of the development cycle of videogames, for beta testing, and even after games have been released to maintain them [61]. Nevertheless, it is important to go through examples in both sides. Therefore, in this section we will be looking at studies done with the intention to help game developers enhance their games, and we will be looking at studies done with players data, with the intention so study their approaches to games, and show them the results of their attempts. Walner and Kriglestein [61] make a differentiation between the two target audiences for which the visualizations are developed, additionally they identify that depending on the target audience, different representations should be considered. In their work they identify the five techniques that they consider the most important, charts and diagrams, heatmaps, movement visualizations, self-organizing maps, and node-link representations. We will be going over three of those representations, since they were at one point deemed the most important for our research.

Charts and diagrams are useful in the case where specific questions require answers but exploratory data analysis is not suited for the case. These are very commonly used in gameplay analysis tools in order to present quantitative data.

For example in Figure 3.3 we can see the Damage Per Second (DPS) meter for a raid fight in Guild Wars 2 (GW2) an Massive Multiplayer Online Role Playing Game (MMORPG) devel-

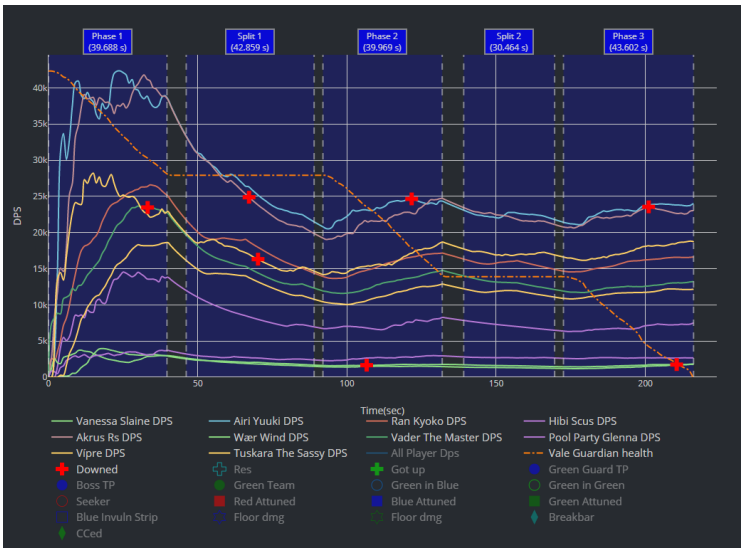


Figure 3.3: Damage Per Second (DPS) chart for Guild Wars 2 (GW2) to analyse player performance generated through the use of arcdps [54] and seen through dps.report [39], both third party programs

oped by ArenaNet [3], these charts can be used to understand and analyze player performance throughout the fight, but give little insight on player intention.

Heatmaps are especially useful to present information about gameplay metrics to players when they can be mapped into specific coordinates. Through these maps, teams and players can become aware of specific areas of danger to avoid, or where they should be more careful. For example certain areas of maps that have a bigger concentration of all deaths in said map, this can be seen in Figure 3.4 where most of the deaths are gathered in 4 to 5 areas of the map. Depending on the games, this can also be educational by carefully mapping helpful information, such as default ways to defend or attack parts of the map in competitive shooters that fall into a pattern.

Movement is a vital part for many 2D and 3D video games since traversing maps is a necessity in many games. Having this in mind **movement visualizations**, are great to analyze how players interact with the maps and how they choose to traverse them. These can then be used to try and understand player intention or to access decision making from players when paired with events that games can log. We can see an approach in Figure 3.5 where a battle map is created in order to visualize the movements done by both teams in a match of World of Tanks (WoT) [63][62].

We only mention these three methods of representation from the work of Wallner and Kriglsteint [61] since they are the most important for our thesis.

When it comes to playtesting data, we can differentiate between tools and studies which are focused on pre-released games, and studies and tools done to upkeep live service games. For example, Wallner et.al. [64] used Infinite Mario as a case study to show the usefulness of the two data types which they identified, objective in-game data and subjective player data (Figures 3.6 and 3.7). Through this process the research team identified three issues with the visualization of playtesting data. First the sheer quantity of data collected needs to be efficiently analyzed

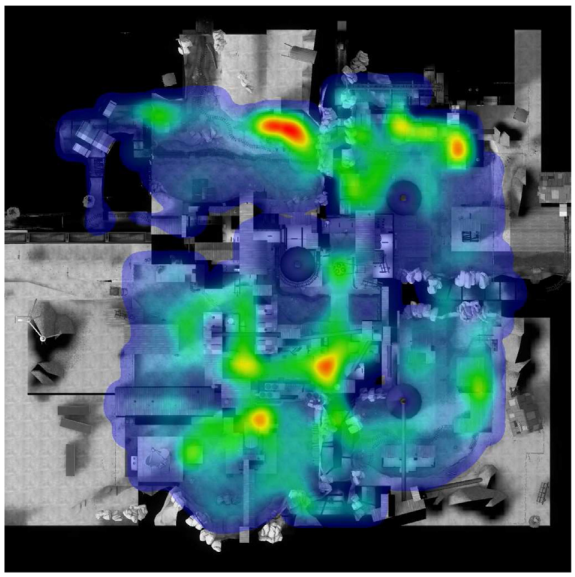


Figure 3.4: Heatmap of player deaths in the map Dustbowl from the game Team Fortress 2 (TF2) [61]

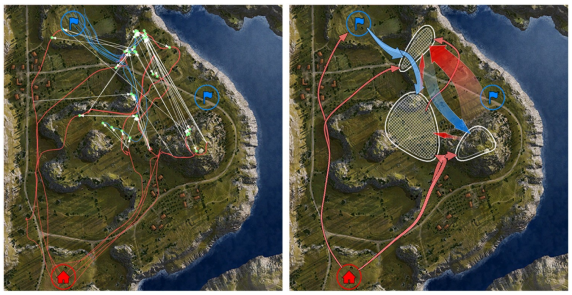


Figure 3.5: Battle maps and the original data derived from replay data on map Cliff on the game World of Tanks [63][62]

and understood. Second playtesting relies on multiple data sources and datasets which need to be integrated and tied together so that we can advantageously use them. And third in order to facilitate interpretation of data, developer backgrounds have to be taken in considerations when assimilating data [64]. In their research their main contribution was achieved by proposing an aggregated visualization which uses multiple aggregation techniques, such as, clustering, territory tessellation, and trajectory aggregation. This was done, in order to effectively assist developers in utilizing playtesting data. And as a second contribution, they provide an understanding and a supporting argument in when aggregated or non-aggregated visualizations are the most appropriate to use [64].

When it comes to tools which are more focused on live service games, Medler et.al. [38] refer to the type of game analytic which can influence how games are designed through player analysis. They proposed that, by monitoring player behaviour and logging in-game events or metrics such as starting levels or specific movement actions done by players they could determine how players play their games. In continuation, the data could be used in a game design point of view,

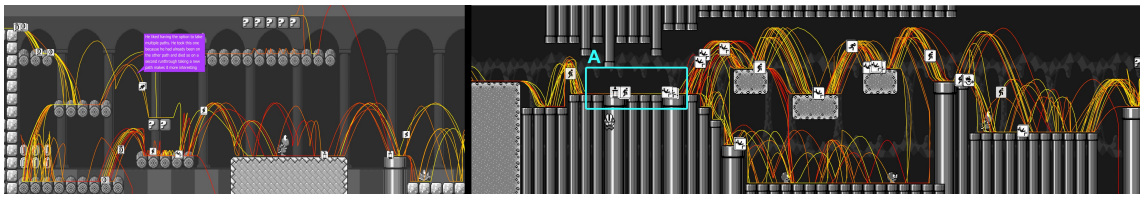


Figure 3.6: Individual player trajectories are visualized using color-coded connected line segments with color indicating arousal by Wallner et.al. in [64]

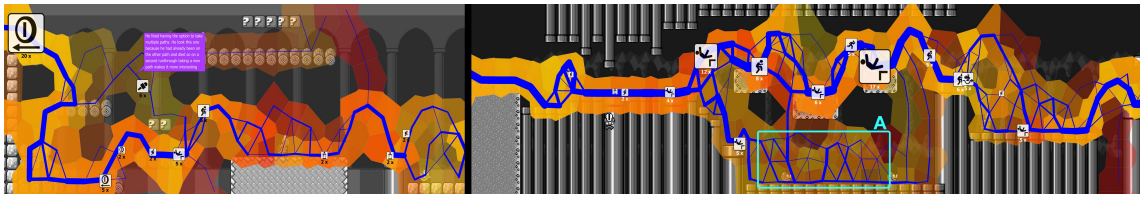


Figure 3.7: Aggregated visualization of the data depicted in Figure 3.6 by Wallner et.al. in [64]

since, behaviour data is useful for determining how players are actually playing the game in their natural environment, versus other evaluation methods such as self-reporting surveys or controlled playtests. In order to achieve this, they developed a visual analytical tool. A tool they named Data Cracker 3.8. Their intentions with this tool, were to explore how visual analytical practices based on player behaviour could augment the game design process, and to use Data Cracker as a case study for evaluation. Data Cracker was a prototype developed along with Electronic Arts (EA) as a part of the companies initiative to provide their game studios with analytical tools. However, development of the tool was not only meant to be an experiment on how to best create analytical tools, but it was also a chance to explore the difficulties that can arise when analytic tools are inserted into a creative process such as game development. With this study, they intended to present the insights gained from working with Dead Space 2 (DS2) and analyse how the development of a visual analytic tool can shift throughout the game and team changes [38].

From the development of Data Cracker there were multiple insights gained. These could be divided into three different categories, production, functionality, and game team integration. Throughout the production process, they realized that building a tool in parallel with the game development cycle and having early visualization prototypes would familiarize the developers of DS2 with Data Cracker. By displaying visualizations that offered different levels of detail team members were not overwhelmed with too much data at once, and this helped the tool to be identified as a part of the DS2 team and created further interest in it. For Data Cracker's functionality a system was built that aggregated data across multiple dimensions including time, in-game maps, and gaming platform. This meant that the tool was quick to handle queries which further increased the ease of access to data by team members. Analytic tools, like Data Cracker, that have the abilities to monitor millions of players after a game has been released and report data to game developers, for which otherwise they would not have access to, are something that should be discussed and considered when pursuing the creation of a live service game, in order for the

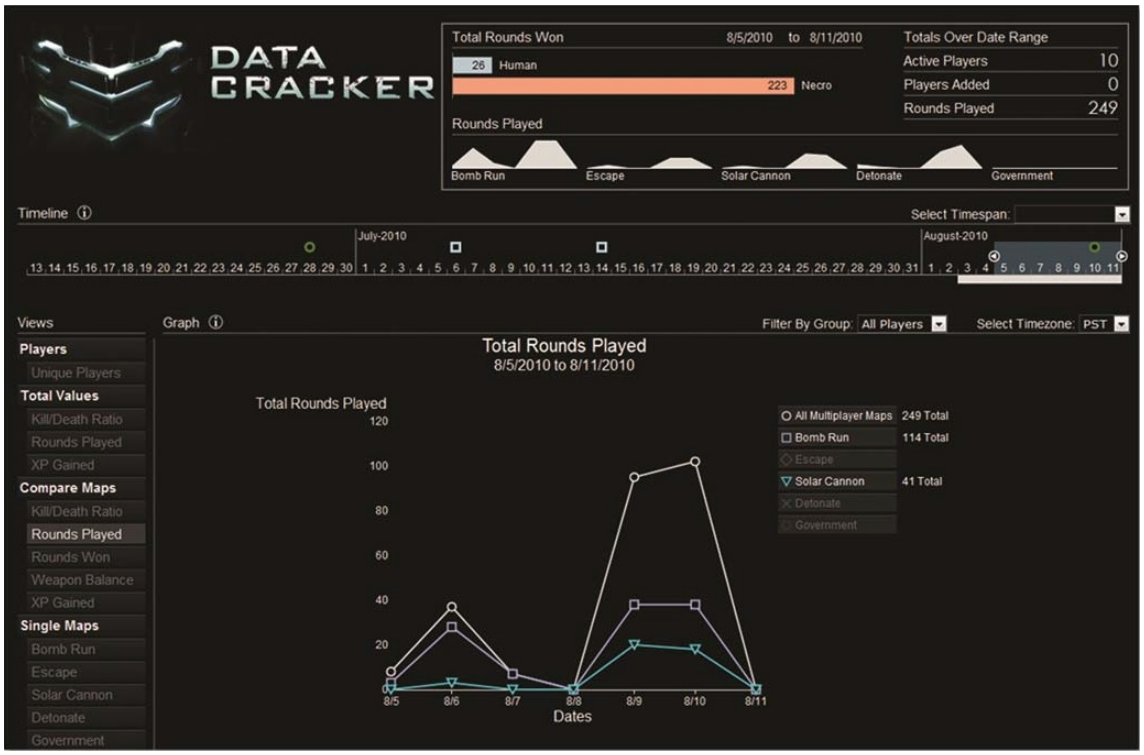


Figure 3.8: Screenshot of the Data Cracker client-side website by Medler et.al in [38]

developers and game analysts to have a bigger disposal of tools at their use [38].

3.4 Summary

Through this chapter we analyzed multiple studies done and saw the importance of game data science, along with telemetry data and the proper visualization techniques. In this chapter we saw multiple studies that analyzed and presented tools to either facilitate or enabled data recovery and data analysis. These tools were used with multiple intentions, either to test visualization techniques and present data to study users, or to facilitate the development and balance process for the game developing companies that want to maintain and support their live service games in a faster and more data driven way.

Chapter 4

Building a Dataset for FPS

In this chapter we will go through the work done towards the attempt to create a game data dataset for a First Person Shooter. In Section 4.1 we will introduce the decisions made and the reasoning on why we chose to use Counter Strike: Global Offensive (CS:GO) as the game to obtain data for. In Section 4.2 we will go over the existing third party applications that can be used as auxiliary tools by players. In Section 4.3 we will present the tools used in the work loop found to obtain game data for CS:GO matches, we also go over the steps needed to obtain match data for a player along with the restrictions found. Finally in Section 4.4 we present the reasoning behind the switch to Valorant and what effort was made to obtain data for that game and why we moved along and dropped this approach.

4.1 Game Selection

When it came to selecting a First Person Shooter, we had to consider which games we could use for the process of data gathering while focusing on games that allowed for automatic, fluid and non obstructive data gathering methods. Initially when the process was explained by the Professors, the games that came to mind were Counter Strike: Global Offensive, due to its popularity and a more linear game play. Since the maps are static and there are no abilities, the game would be the most linear to consider for study. We also had considered Tom Clancy's Rainbow Six Siege, a game where there is map destructibility, and every character has their own unique abilities along with their own set of weapons, which added a massive lair of complexity granting it to be dropped. And finally, we considered Valorant, a game where the maps are static and the weapons are also available to every character, much like CS:GO, but each character has their own set of abilities which adds some complexity, but it ends up being bearable. In the end we went with the option of using CS:GO and if needed we would change to Valorant as a last case scenario.

4.2 Third Party Tools for FPS

When it comes to third party tools for games, there are several tools that gather data from matches and present it to the players. This data can be used in order to help them not only get better at

the games they play, but also present players statistical data about their performance, either for the round they played, or for the whole game. Outside of the context of the FPSs there are also tools that guide players throughout matches. Having this in mind the tools found were the following: *blitz.gg*[1], *lolalytics.com*[2], *op.gg*[3], *csstats.gg*[4], *insights.gg*[5], *tracker.gg*[6].

4.3 Tools for CS:GO Data Gathering

To be able to obtain data from any of Valve's games, we have to mention the platform they originally developed with the intention to distribute the games they developed, *Steam*[7] [35][65]. In the current days, *Steam* is the biggest platform for digital game distribution, and is also one of the only ways we can interact with the games developed by *Valve*, in our case the Counter Strike series of games.

In order to obtain game data for CS:GO we had to utilize several third party and *npm* packages that could be used in future iterations in case the game were to be used. In this section we will go over some of those tools and what role they had in the workflow found for the Python script that was used to obtain data from matches. So the tools found are the following:

- *Steam python module* [59] - an *npm* module which is a reverse engineering of Steam which is used to emulate sesions;
- *CS:GO python module* [58] - an *npm* module which is a reverse engineering of CS:GO utilized to communicate with the Game Controller behind the scenes of CS:GO;
- *awpy python module* [46] - a third party tool that can be used to turn CS:GO match replay files into event datasets.

To collect information from CS:GO players from the *Steam* platform, it was necessary to create a Steam account. This step was done so that we could interact with other steam accounts in order to gather their match data. The account created to interact with the *npm* modules was *fcul_csgo*, and then through the use of my personal account *Petinga*, games where played to be used as trials for data gathering. But before any match was played there where two other steps needed to be done, first through the use of my personal account we had to create a Match Sharing Code [52].

Following this, the account from which we were going to acquire data needed to obtain an API Key for Steam through the Steam Community page [51]. This enabled the account to request the steam API and CS:GO for match codes. To obtain match codes from any account we require a couple of codes from that same account, these would be:

[1] `https://blitz.gg/`
[2] `https://lolalytics.com/`
[3] `https://www.op.gg/`
[4] `https://csstats.gg/`
[5] `https://insights.gg/`
[6] `https://tracker.gg/`
[7] `https://store.steampowered.com/`

- *The Steam Page Uniform Resource Locator (URL)* - the accounts community page URL;
- *The Match Sharing Code* - the code created previously on the account through the Steam Community Page;
- *An old known CS:GO match code* - a code for a match played by the account from which we will gather data, these can be obtained through multiple options, but the simplest is from CS:GO.

We must also have in mind that we will only be able to access matches that happened after the match codes given. Once we obtain the three codes, we can then get the players steamId through the use of the tracker.gg API, for this we will require a tracker.gg account and an API Key to submit requests. For these requests we will need the players **steam user community page URL** and the **platform** they play on, which by default is steam. After the implementation of the workflow, we came back to this step. This was due to the fact that there should be an easier way to obtain the **steamId** of a player. From this request we can obtain the users **steamId**.

After some research, we concluded that in order to do things in a more fluid workflow the solution we presented previously, is acceptable, but it requires the codes from the players. However, since we already request players for information, we can also request that users also deliver us their **steamIds**. Since, they can easily acquire them through their account detail page, in the Steam application.

Once we have the steamId, we obtained the required information to request match codes from the Steam API. This process can be seen in detail in the Valve Developer Community Page [9]. Once the codes have been obtained and saved in the preferred format we would then have to use the **Steam Python module** [59] and the **CS:GO Python module** [58] to decode them. This is done through the use of the **csgo.sharecode.decode** function from the CS:GO module, the match codes then return three values: *matchId*, *outcomeId*, *token*. With these three values we can then use the **request_full_match_info** function to get the match information, from which we just want the **match sharecode**. This information does come in a *Protomessage* format, which we then turn into a dictionary, and then dump into a JavaScript Object Notation (JSON) file.

From the JSON file we can obtain some basic information on the match played, more specifically we can see the state of the leaderboard at the end of each round. But this is of little importance since there is little usable information that can be obtained from those states. Although we can also obtain a link to download a zipped replay for the match. After unzipping the replay it can be seen from the CS:GO application but it also holds the information for each tick of the game. To obtain this information we would need to use a **demoParser** on the demos. There are several parsers that can be used, but in our case we used **awpy** [46].

4.4 Discussion and Conclusions

After reaching the conclusion that CS:GO was not feasible due to the fact that we would be dependant on players to then obtain player data. Therefore we reached the conclusion that this would not

work for our timespan but could possibly be used in a future project if the data gathering started fairly earlier. Due to this conclusion, we moved towards attempting to use the same approach to another game, more specifically to Valorant. Where, through the use of the Riot API we could facilitate the data gathering approach. Unfortunately, due to the fact that to get data on Valorant you require a Production key, the process was fairly hindered. A request for the Production key was submitted through the means indicated in the Riot Developer Portal. For this request, we had to submit some mandatory information such as a product name, a product description, and a product URL. There is also a strong implication that a prototype of the app has to be hosted in the URL given. This is so that Riot employees can test and approve the prototype for a Production key. Due to the non existence of the prototype at the time of the request, we also abandoned the possibility of using Valorant as the game for the project, and moved along with LOL as the main game for the rest of the project.

Chapter 5

LoLChrono - Analysis and Design

LoLChrono is the prototype web application we developed in order to support and enhance the past applications, developed by our colleagues by filling in a gap that was left in their iterations. The previous application we are mentioning is VisuaLeague. This application was developed throughout multiple iterations by Carreiro [7], Vieira [60], Moucho [42], and Afonso [1]. The past application had its focus on the analysis of player's matches, and their performance on those same matches as a singular unit and as a team. Our intentions, therefore, were turned towards a different type of analysis. We turned our sights onto executing player statistical analysis, focused specifically on a time analysis of players in terms of playtime. For this the MOBA genre was the best option and therefore we looked once again towards League of Legends. Throughout this chapter we look at the work done in order to develop the application, and the choices that influenced the development of the application. In Section 5.1, we will go over the description of the application along with some little notes on the design choices for the applications layout. In Section 5.2, we analyze the architecture choices made for the web application, and the evolution that these suffered throughout the development period of the application. Additionally we also introduce the several parts that compose the application with their responsibilities. In Section 5.3, we present the technologies used, and how they helped the development for each of the main parts of the application. In Section 5.4, show the multiple APIs used in the gathering of the information on the players selected, and we also go over the specific endpoints requested to do so. In Section 5.5, we present the multiple design choices made for the multiple views existent in the application. In Section 5.6, we will be going over the testing done to the web application, the results obtained and the knowledge inferred from those results. And finally, in Section 5.7, we will summarize all the work done for the web application.

5.1 Prototype Description

LoLChrono is a web application with which a user can search and visualize information related to their own, and other players, playtime in the popular MOBA game, League of Legends. The main goal with the development of this application was to present time based statistics, such as total time played, average time played per match, average time played in the multiple positions of

the game, and others, to the playerbase of LOL. Our intention with this was to show the players their habits of playtime, in a way they could understand. These same time statistics could also be used to study player profiling and help characterize the player behind the screen better. Besides this, we also considered the possibility for there to be a link between playtime and the position on the ranked ladder at which the player is currently stands on. And if that same position would influence the players to play more or less.

League of Legends was the game we choose for multiple reasons. First of all it is the most popular MOBA, at the time of writing the thesis it is also a game that is fairly beginner friendly in terms of difficulty curve, and it is a game that does not require a very powerful Personal Computer (PC) in order to play. Therefore it is quite easy for any person to at least attempt to play the game, making it a very good case study. Besides this, given that the population of players of League of Legends is quite massive, finding a good player group to test the prototype on, could be easier than other options in comparison. Besides the previous reasons, the API maintained by Riot, the developers of LOL, is of simple access when it comes to LOL which also influenced greatly the use of LOL as the target game. Although the original project idea was to move away from the MOBA game genre, as seen in Chapter 4 ,we ended up coming back to it due to its ease of use and availability.

Past projects had a major focus on player performance, both as a single player and also as a unit within a team [1], which made us take a different approach. As mentioned before we focused more on the playtime information for a player, and how ranked disparity could influence the playtime of a player. Unfortunately, the Riot API does not save rank history throughout time, so we had to focus on creating a tool to analyze a season of data from a player and present a visualization tool that could be used in order to more easily interpret said information. In order to have the ranked information for a player, there would have to be a continuous interaction with the Riot API in order to create a dataset with the ranked info of players throughout a season of LOL. We could then correlate that information with the playtime information kept by Riot.

The application is composed of three main pages. These are the Main page (Figure 5.1), the All Players page (Figure 5.2), and the Player page (Figure 5.3 and Figure 5.4).

The Main page (Figure 5.1) which can also be considered the Home page, presents the user with a search bar that is used to search for a player given its summoner name and the region in which it plays.

We also have the All Players list page (Figure 5.2), which shows all the players that are in the database along with their summoner name, their summoner region, their summoner icon, and the number of games that they have in the database. It is important to note, that the database might have less games than the total additive number of every player combined, since two players might have played the same match, but it is counted once for each player. Besides this information, the All Players list page, also has a mock-up feature to add players to the database, currently it does not do anything but emulate a working feature. This is due to the process of adding a player to the database being fairly time consuming due to the restrictions from Riot and their access rules for

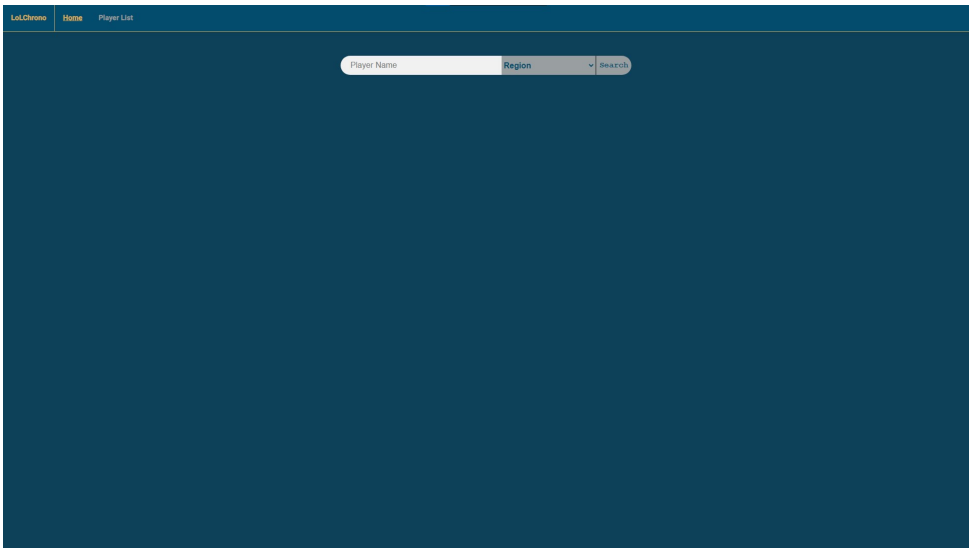


Figure 5.1: The Home page of our prototype

their API, which is explained in Section 2.2.3. This could be fixed if in a future iteration a higher level API key with less strict access restrictions was acquired.

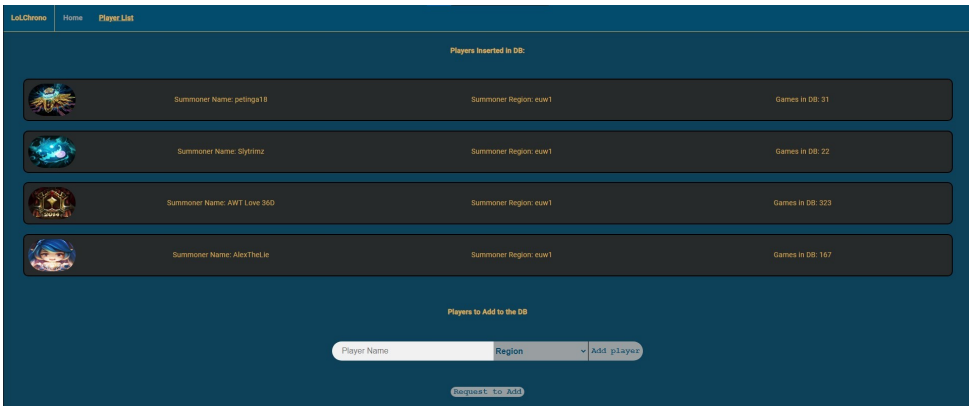


Figure 5.2: The All Players page of the prototype

Finally, we have the Player page (Figure 5.3 and Figure 5.4) where most of the information is shown. In this page, we show the current player information, summoner name, account level, region, and summoner icon. And for each player we show two sub-pages, in the form of tabs: Player Information, shows the temporal data for the given player, through the use of several graphs as seen in Figure 5.3, and the Player Matches shows all the matches that the Player played in for the given time period we studied in a list format with some basic information for each match as seen in Figure 5.4.

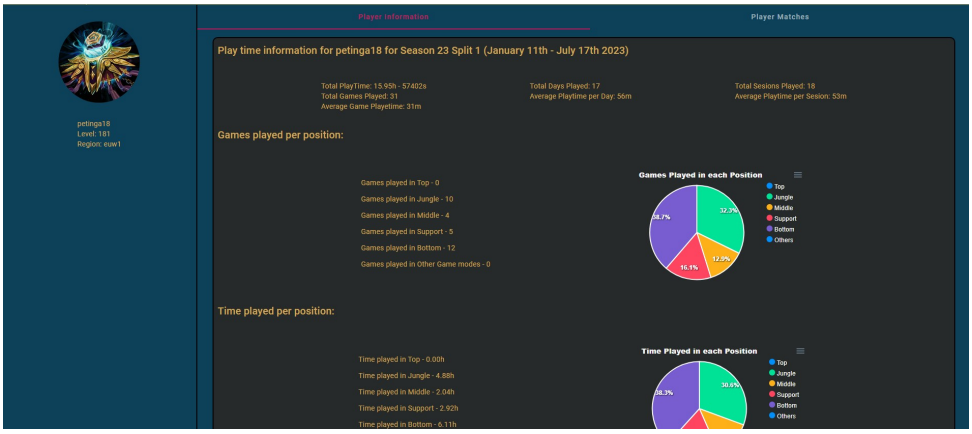


Figure 5.3: The Player Information Page for player petinga18

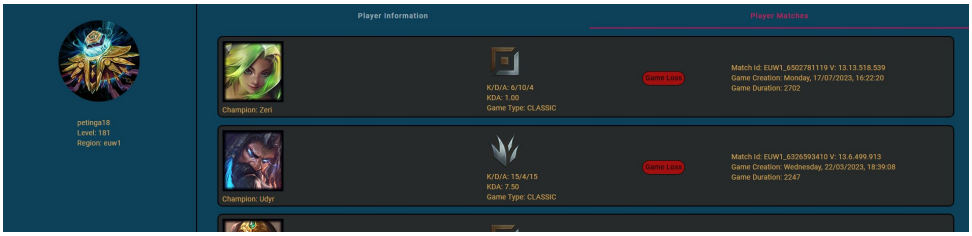


Figure 5.4: The Match List Page for player petinga18

5.2 Prototype Architecture

In this section, we present the architecture of the prototype. In an initial approach to the architecture of the prototype, there was a discussion with the supervisors and from it we reached the conclusion that due to the past experience with the tools and their popularity for projects similar to this one we would be using Angular [26] for the frontend part of the prototype, and Node.js [15] along with Express [14] for the backend. Bearing these technologies in mind, we thought of using the Model-View-Controller (MVC) architectural design pattern to base the application on, but with the variation of making Angular be responsible for all the Views, and having all the logic from the Controllers and Models be handled in a back-end server created using Node.js and Express. This logic can be better viewed in Figure 5.5.

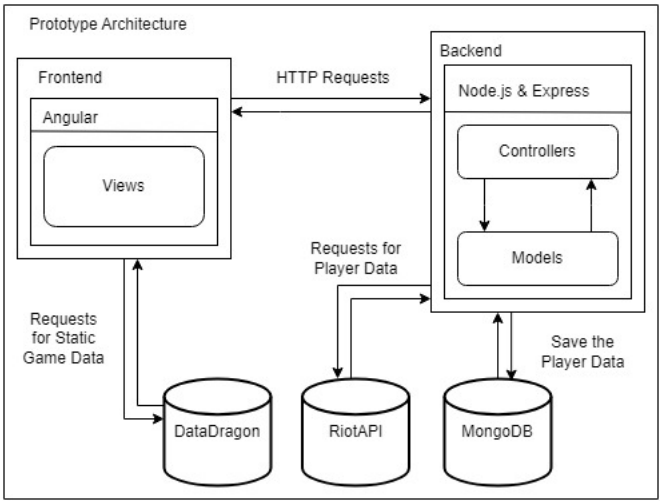


Figure 5.5: The initial Architecture thought out for the prototype

As the development went on, the final architecture for the prototype suffered some changes. Although nothing major was changed, some logic had to be added to both ends that was not foreseen in our initial assumptions, therefore we ended with the architecture that can be seen in Figure 5.6. In this figure we can see the architecture in more detail, with every major part of the prototype being represented.

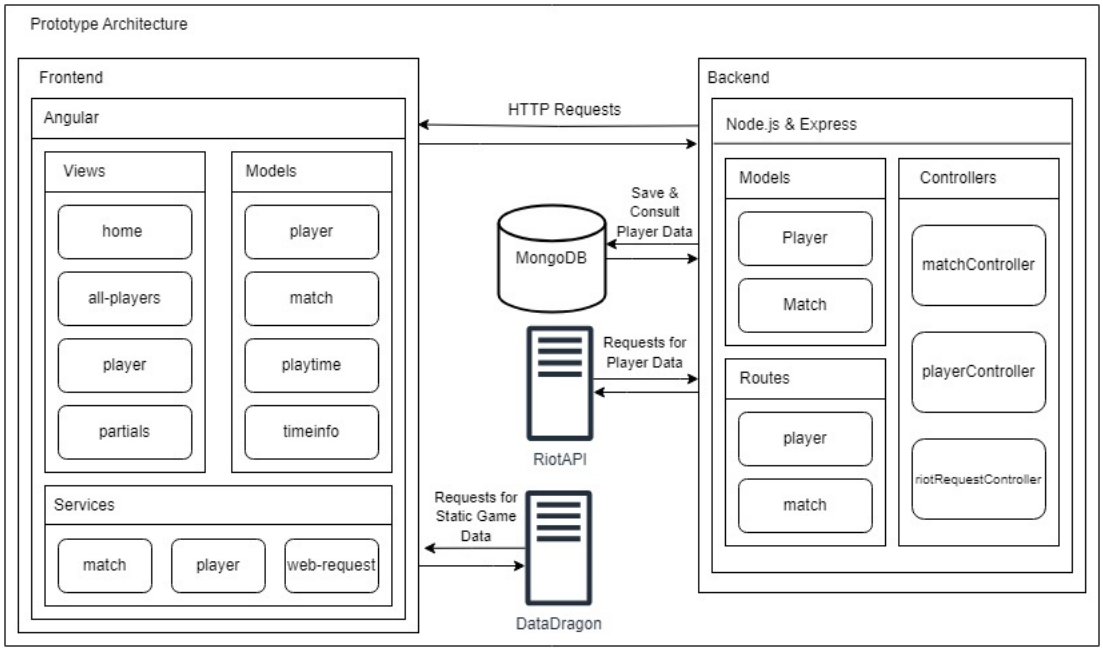


Figure 5.6: The final Architecture considered for the prototype

In order to better understand the architecture, we present the components of the prototype. We start by discussing the components of the frontend, and then go over the backend components.

5.2.1 Frontend

The frontend is composed of three main components which are the **Views**, the **Services**, and the **Models**. There are also the APIs with which the frontend communicates with. But these and the other APIs used for the project will be addressed in Section 5.4.

Views

The Views represent the multiple pages that compose the frontend of the prototype. These are the Home page, the All Players page, the Player pages, and the Partial pages. The Views, are also responsible for presenting the information of the application to the users.

The **Home** page is responsible for being the main page for the application, it has a search bar that can be used to search the database for players in order to access the page with their players' information.

The **All Players** page is a page that can be access through the header of the website. In this page we can see all the players that are in the database, besides the player list, we also have a mock up function in this page that emulates adding a player to the database.

The **Players** page can be accessed through the search bar in the Home page, by using the summoner name and region of a player. In this page, we can access all the details related to a specific players. This involves the information of that player, the statistical information that can be seen through multiple graphs, and the match history for the player. We should also bear in mind, that the statistical information and the match history are two separate pages. These can be accessed through a tab available in the Player page, seen with more detail in Section 5.5.3.

The **Partial** pages are views that can be, and in some cases are, used in multiple places throughout the application. These are the search bar, the match info, the header, and every chart created. They are responsible for multiple secondary jobs in the prototype.

Services

The Services in the application, are responsible for submitting requests to the back-end, in order for the pages to have the necessary information that needs to be presented to the users.

The **match** service is responsible for submitting requests that have to do with the match details. There is a single request in this service, which is to request the details of a match for which we have to use the match id.

The **player** service is responsible for submitting requests that have to do with the player details. There are several requests in this service. The first request returns all players, and we then have three others which return data for the player with the given player id. From these three, the first request returns the player information, the second returns the time statistics for the player, and the final one returns the daily playtime information.

The **web-requests** service is the final service for the frontend. This service is responsible for sending the requests to the backend. While the previous services build the requests, they are

dependent on the **web-requests** service, to send them to the backend and return the data from where it was requested.

Models

The Models for the frontend side of the application exist simply for the creation of classes that can be used to receive data in order to create object like structures. These are the **match** model, the **player** model, the **playtime** model, and the **timeinfo** model. These are all just classes which represent these concepts as Objects in order for them to be populated throughout the frontend when we need to do so.

5.2.2 Backend

The backend main components are the **Controllers**, the **Models**, and the **Routes**. There is also the APIs and the database with which we interact, but these will be addressed in Section 5.3 for the information on the database, and Section 5.4 for the APIs.

Controllers

The Controllers in the backend are responsible for having the biggest part of the logic within the prototype. The main task that they manage is the communication with both the Riot API, and the database. These two tasks are vital for the overall correct operation of the prototype. In total we have three different controllers in the backend these being; the **playerController**, the **matchController**, and the **riotRequestController**.

The **playerController** is responsible for accessing the database and obtaining the important information for the frontend, it deals with player information, and playtime information.

The **matchController** is responsible for accessing the database and retrieving the match data.

The **riotRequestController** is responsible for gathering information from the Riot and inserting the information in the database.

Models

The Models for the backend are responsible for creating the Object structures that will facilitate the insertion of the data in the data base by using mongoose Schemas (explained in further detail in Section 5.3.3). There are two models in the backend section, and these are the **player** model, and the **match** model.

Routes

The Routes for the backend are responsible for receiving the requests from the frontend. Then they send them towards the part of the backend that will gather the necessary information to respond correctly. Two separate routing files where created, the **player** router which handles the requests for player data, and the **match** router which handles the requests for match data.

5.3 Technologies Used

In this section, we will present the multiple technologies used for the development of the prototype. The two technologies suggested by the supervisors were Angular[1] and Node[2] in the application which were Angular and Node. In Section 5.3.1 we present the multiple technologies used for the frontend. While in Section 5.3.2 we analyze the technologies used for the backend. Finally in Section 5.3.3 we introduce the technologies used for the database.

5.3.1 Frontend

The technologies used for the development of the frontend were Angular[26], which was the web application framework used to create the website. Angular Material [27], which was used to facilitate certain parts of the designing since it is a UI component framework. And finally, ApexChart [2], which is a charting library that facilitated the creation of the charts visualizations.

Angular

Angular[1] was the web development framework used to develop the prototypes' frontend. This choice was based on several reasons. The main reason behind the use of Angular for the development of the frontend was the fact that in past projects, we had used Angular, therefore this past experience would greatly facilitate the development of the prototype. We also kept in mind that Angular has a Command-line Interface (CLI) tool that would greatly facilitate and streamline the development process in ways that Vue[3] and React[4] could not. Any of the three tools have their ups and downs, but in the end we picked Angular over the others due to its ease of usability, scalability, effectiveness and fast pace of development.

There was also the fact that the supervisors indicated it to be the appropriate way for the development of the frontend, and with the many features provided by Angular it proved to be the most correct way to develop the prototype. As a platform, Angular is built on TypeScript, a language that builds on JavaScript in order to provide better tools at any project scale. With this said, Angular provides great scalability with a component based framework, a plethora of developer tools which facilitate and streamline the development of applications no matter their scale, and finally with it's collection of well integrated libraries, a wide variety of issues that existed with other choices, were covered by Angular itself.

For the prototype, Angular and their CLI compiler were the adequate tools to guide the development of our small scale application. And several User Interface (UI) components were used along with Angular to streamline the creation of the frontend.

[1] https://angular.dev/
[2] https://nodejs.org/en
[3] https://vuejs.org/
[4] https://react.dev/

Angular Material

In order to facilitate and streamline certain features on the frontend prototype, we took advantage of Angular Material[5], a UI component library. The use of Angular Material came from the necessity to add certain features to the user interface, which would request a good amount of effort to develop in-team. Since we noticed that the same team that maintains Angular, develops Angular Material we considered it to be the best choice to enhance the prototype with, the least amount of issues coming from the integration of the library. Angular Material's library provides multiple solutions for frontend design, such as, date pickers, input forms, slide toggles, and much more.

In our prototype the Angular Material library was mostly used in the player page, to create the tab division in the page.

ApexCharts

ApexCharts[6] is a free open-source chart library that helps developers create charts with the data from their projects. Apexcharts has a wide range of chart types, and focuses primarily on making flexible and fully responsive charts that work on any platform. Besides this, it also allows developers to customize their graphs. ApexCharts can also be integrated with several languages. In our case, we utilized the Angular solutions, but there are also Javascript, Vue[7], and React[8] solutions that can be used.

In the prototype Apexcharts was used to create the charts that display player information. Whether that is, date time information, or playtime information, all charts displayed in the frontend prototype were created using the Apexcharts library.

5.3.2 Backend

The technologies used for the development of the backend are, Nodejs [15] along with Express [14] to create the backend API, and Postman [47] to test and verify the working order of the backend.

Nodejs & Express

Node.js along with Express[9], were the two technologies used to develop the backend prototype. Node.js is a Javascript runtime environment designed to build scalable network applications. The use of Node.js was the most indicated due to it's position in the web development environment. This, along with the fact that it is an event-driven asynchronous environment, made it the most indicated choice for the prototype.

Express is a web framework for Node.js that provides a set of features for web and mobile applications, with a plethora of Hypertext Transfer Protocol (HTTP) utility methods and middleware

[5] https://material.angular.io/
[6] https://apexcharts.com/
[7] https://vuejs.org/
[8] https://react.dev/
[9] https://expressjs.com/

at the disposal of it's users. It was used to facilitate the creation of the API in a quick and easier manor.

Postman

Postman[10] is an API platform for building and using APIs. Postman is used to simplify the testing and streamline the effective creation of APIs. For the prototype, Postman was mainly used in order to test and verify the integrity of the backend API, by testing the routes for the HTTP requests.

5.3.3 Database

The technology used for the creation of the database was MongoDB [40] along with their Graphical User Interface (GUI) MongoDB Compass [41] in order to maintain and manage the database.

MongoDB[11] is a Not Only Structured Query Language (NOSQL) database solution, which utilizes a JSON-like document structure along with schemas, to insert and save the information needed for the application. MongoDB Compass[12], the MongoDB GUI, was used to verify, test, and maintain the databases. Through it we were able to query and test the integrity of the data. These technologies also enabled the creation of indexes and compound indexes, which enabled a more efficient access to the data, which in some cases reduced our accesses to entries by ten fold. With all that was said previously, the utilization of MongoDB, and their platforms, as the database technology was the best, and most efficient, choice for the prototype.

5.4 Data Gathering

In this section we will go over the two APIs used to gather data throughout the prototype. The two APIs used are both maintained by Riot and each serves their own purpose. Riot API [18] is the main API maintained by Riot, from where we get most of our data, while DataDragon [20] is an API used to request for assets that Riot makes available for third party developers.

5.4.1 RiotAPI

Riot API [18] is the main source of data, that is used by the prototype. Through Riot API, we acquire information on the players and their data. We acquire all the game Ids that the players played in the time period we set for the study, and along with every game Id we also acquired, for every game, the game data. In this section, we present the methods used and how the requests were made. One thing we should always keep in mind when we talk about the Riot API, are the multiple restraints with API keys, explained previously in Section 2.2.3.

When we talk about player data requests, we must consider that a certain information must be known from the players previously in order to acquire the Ids for that player, and there are also several ways to make these requests. In our case, when we request the basic player information,

[10] `https://www.postman.com/`
[11] `https://www.mongodb.com/`
[12] `https://www.mongodb.com/products/tools/compass`

we utilize the **Summoner-V4** endpoint[13]. This endpoint has several options to acquire player information, but in this case we use the summoner name of that player along with the region in which the player plays, to then acquire the basic id information for that player. These requests were vital to populate the database with players, through the use of their summoner names and regions we acquired the three ids for the player, *id, accountid, puuid*, along with these we acquire the *summoner name*, the *profileIconId*, the *revisionDate*, and the *summonerLevel*. In order to save this information acurately for the database, we add the *playerRegion* to the JSON response from the request, which can be seen in Appendix A in Figure A.1, along with the description of the data types that are received in the requests response.

Once we acquire the player information, we also want to add a List of the game Ids the player played in their database entry. To achieve this we have a loop of requests to the Riot API. In order for this to happen we use the player *puuid* to request matches from the **Match-V5** endpoint[14], which is the Riot API endpoint to acquire the player's list of *matchids*. These Ids are iterated in 100 matches per request between the time frame of *January 10, 2023, at 23:59 British Summer Time (BST)* to *July 17, 2023, at 23:59 BST*. This time frame is the equivalent of the ranked season of 2023 split 1 [29][32]. Once the request loop ends, we append the information to the player's JSON object in the *matchIds* variable so that we can then save it on the database.

The final request we make with the Riot API, is to acquire all game data for each match our players played. In order to do this, we go and for each of the players in the database, we extract their match list. Then for each unique *matchid* in the list we make a request to the **Match-V5** endpoint, which is the endpoint in the Riot API that enables us to acquire match data, to gather all the static information for every unique match in the player match lists. These will then be added to the database, if they were not before, in order to reduce the amount of requests sent to the Riot API.

5.4.2 DataDragon

DataDragon [20] is an API provided by Riot to centralize game data and assets, all of which can be used by third-party developers. These resources can be downloaded into a local folder, or accessed like a traditional API, but with no requirement of an API key. There is also something to have in mind, these resources are manually updated and maintained, just as previously stated in Section 2.2.3. Therefore, they can take a while to be updated and the requests should be updated with every new game patch, due to the fact that new assets can be added to the game. In our case, since we looked for data from a past patch, we could utilize a request for the patch at the end of the season and there shouldn't exist any data consistency errors, unless the users utilize new profile icons which did not exist at the time.

In our specific case we utilize DataDragon to get mostly icons. For example the lane icons used in the matches list part of the player page were downloaded from DataDragon and are then

[13] `https://developer.riotgames.com/apis#summoner-v4`
[14] `https://developer.riotgames.com/apis#match-v5`

used depending on the lane the player is playing in that match. Besides this, we utilize DataDragon to acquire player icons, and champion icons respectively through the following calls:

- *https://ddragon.leagueoflegends.com/cdn/[version]/img/profileicon/[iconNumber].png*;
- *https://ddragon.leagueoflegends.com/cdn/[version]/img/champion/[championName].png*.

These calls respond with the image assets that are then used in the frontend directly.

5.5 Design

In this section, we will present the design decisions made when we were developing the prototype, and we will give the reasoning used for our choices.

First, in Section 5.5.1, we introduce the overall design choices for the prototype, this will include the structure of the web app and its implementation. Then, in Section 5.5.2, we explain the choices made when the all players view was added and we will talk about it's uses. In Section 5.5.3, we analyze the design of the player page, and why it is split in its parts. In Section 5.5.4, we show the reasoning behind the addition of the pie charts to the Player page, and why we believe they were a necessity to convey information. In Section 5.5.5, we explain the design choices behind all of the non pie chart visualizations, and why they were added into the Player page. Finally, in Section 5.5.6, we will go over the decisions made on why the match list was added and why it shows the information it shows.

5.5.1 Overall design

Our prototype was created having in mind two concepts, the first was, that it was a tool that would help players that played LOL and therefore it should try and remind the player of the game. This was the main reason we decided to use the color pallet we used of blue, yellow and grey as the main colors, just as League of Legends. The other concept was that it should not be hard to adapt this tool into a bigger project, or another application that was already developed, or even into a new application that would be developed in the future. Keeping that in mind, we tried to make the prototype in a simplistic way that could easily be altered and iterated upon to make it fit into different tools. We ended up with a simplistic prototype with a minimalist design, composed of three different pages which are, a Home page with a search bar which can be used to access player data (Figure 5.7), the All Player list with the full list of players in the database (Figure 5.8) and finally the Player page where we can analyze the multiple graphs with playtime data and see the match history for the players (Figure 5.3 and Figure 5.4).

This is also why the prototype uses a simplistic way to navigate and an easy way to acquire the data for any player in the database. By using a search bar along with a dropdown, we can easily adapt this search method to any other application that either uses the same search method, or is made with the Riot Sign On (RSO) login method made available through Riot [21]. We did not use the RSO method since it requires a production key which we did not acquire. In a future

Figure 5.7: The search bar in the Home page of our prototype

implementation, if the tool continues to evolve, acquiring a production key and getting RSO login into the prototype would greatly improve it's responsiveness and uses, since it would most likely be able to gather data in real time which would greatly increase the prototype's abilities.

Besides acquiring player data, we can also view the full player list that is currently in the database. We can navigate between the home page, which has the player search bar, and the page with the full player list through the use of the header.

5.5.2 All players page

In the navigation bar the Player list option (Figure 5.8) contains the list of players and along with it a mock up function which is not functional, due to being a high time consuming action depending on the player we add to the database. This is only an issue because the current API key locks us at a very slow rate of requests to the Riot API. Currently we can only execute 100 requests every 150 seconds, and with those rates of transactions it is not feasible to make a responsive application that can gather data in real time. If we were to get the best key, a production key, then the rates would be more than fiftyfold what we currently have. And in that case, we could make this mock up function work in real time, although in some cases, depending on how many players we request at once, it could still take longer to gather the data then we wish for it to take.

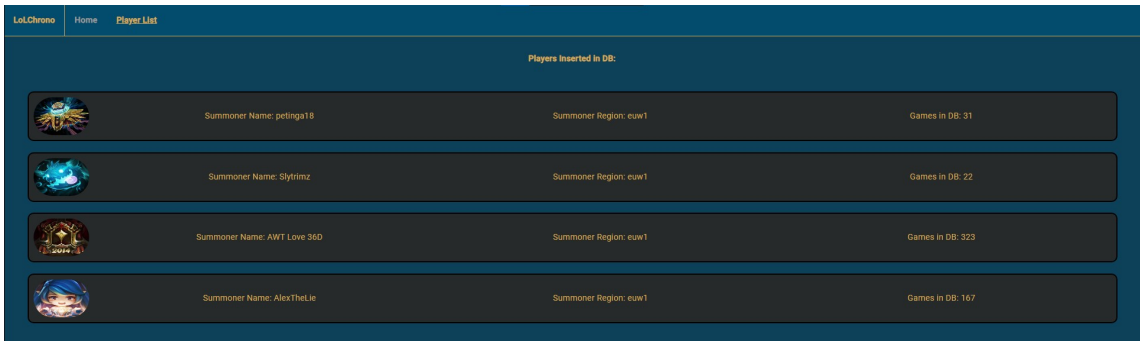


Figure 5.8: The players list in the All Players page of the prototype

The information we found vital to show in the Player list page, was the minimal amount of information, with a specific focus on the information that mattered to identify each player. Therefore, as can be seen in Figure 5.8, in the Player list we show the player icon, the players summoner name, the region they played in, and finally the amount of games they have in the database. It is vital to note, that a player can change region, therefore we could have the same

player in two different regions in a season. In our case we did not have this shown with the Player list.

The mock up function to add players to the database, as can be seen in Figure 5.9, works in the following way. We utilize the same form as the search bar, to request for a new player to be added to the database. After we hit the add player button, a new entry on the player list under the search bar will be created. This entry, will contain the information given in the form, along with a button to remove that player from the list. This was added in the case that a mistake was made when filling the form, and a player needed to be removed from the list. After all players that are going to be added to the database are in the List, we can click the button under the list and request the prototype to add those players to the database. This will activate an alert on your browser, that will read out ”Players added to the database” after this the list will be cleared.

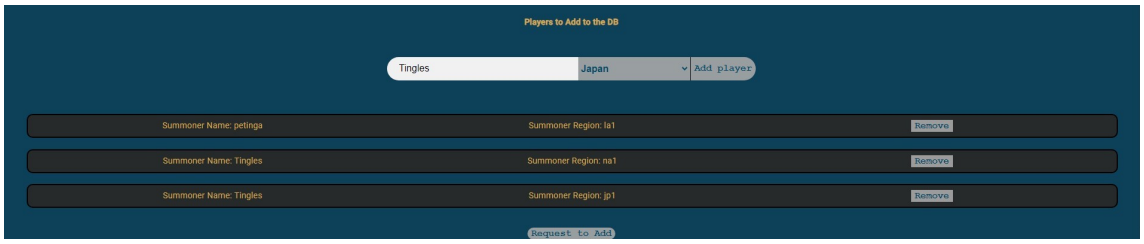


Figure 5.9: The mock up function in the all player page

There is a small issue with the mock up function, which could be easily fixed if we actually contacted Riot API when we add a player to the list of players that we want to add. This issue is, that if we have multiple players with exactly the same information, once we delete one, the other will also be deleted. If we were to contact Riot’s API when we fill the form for adding the players, we could immediately verify if the player exists, and depending on that, associate some information to that entry on the list, making it a unique entry. And in case we submitted a request to then add it again, a check could be made on the list to know if he was already going to be added. This issue was in the end seen with lesser priority due to tasks with greater importance still needing our attention.

5.5.3 Player page

If we were to strip the player page from all the statistical information down to its bare bones, we would have a page that would simply represent the player we searched for, just as it is seen in Figure 5.10. In this representation of the player page, we can see basic information on the player, such as, the summoner icon, the summoner name, the player’s summoner level, and finally the region in which the player plays. It is important to note that the left hand section where the player information is, was deliberately kept rather empty in case more information was added to the player, for example if we had rank information for the time period of information being shown, the ranked information could be displayed along this section. Along with this information, we can also see the two tabs that will display the players information for us, the Player information tab,

and the Player matches tab.

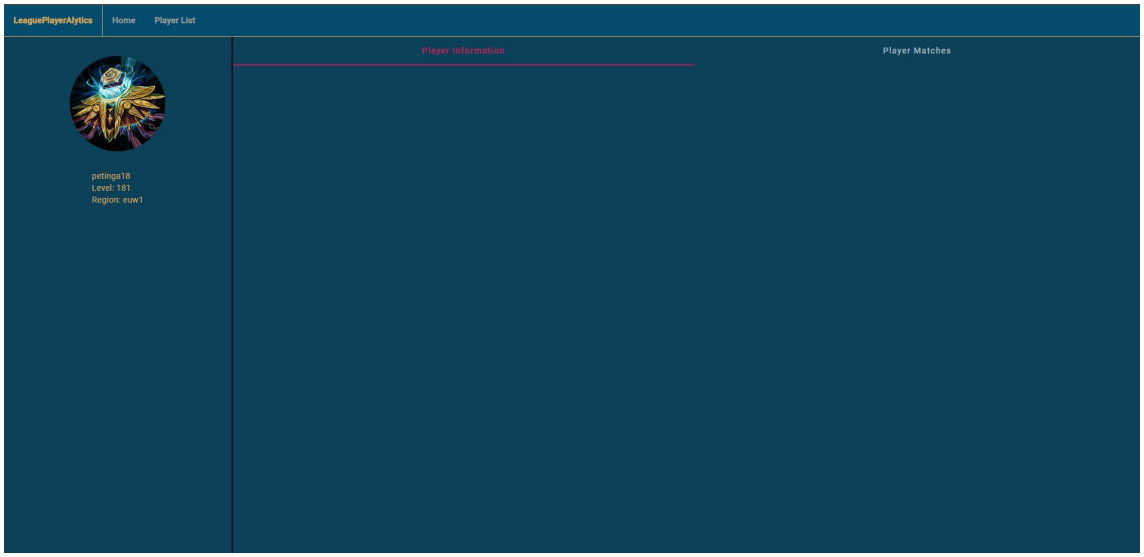


Figure 5.10: Representation of the Player page stripped of all statistical information

Regarding the tabs, depending on the tab selected we will fill the right handed section of the page with the information relative to that tab. The player information tab will show the statistical information relative to the matches played by the player in the first split of the season of 2023 along with the graphs. While the player matches tab, will show all matches played by the player in the first split of the season of 2023, along with some information relative to the matches. Both these views will be seen in further detail in the following sections.

5.5.4 Pie Charts

Before we consider the pie charts, we should take notice of the information that is presented prior to them. In the first section of the player information tab, we represent some statistical information that is important to note (Figure 5.11). First, in the title of this small section, we present the time period for which all information was acquired. This period is the first split of the season of 2023, which represents the time period between January 11th of 2023 up to, and including, July 17th of 2023 . Along with this information, we also convey the total play time for the player both in hours and in seconds for a more exact value, then we have the total number of days where the player played at least one game, the total number of sessions the player did, where a session is defined by when there is a pause of an hour and a half between two separate games, the total number of games played, the average playtime per day, the average playtime per session, and the average length of a game for this player.

Following the information, that is represented in plain text, come the two pie charts that were represented. Both of these represent the matches played in each position, but they are different from one another due to the way they count up the time played. While the first pie chart, seen in Figure 5.12, represents the games played in each position and the percentages of games played

Figure 5.11: Representation of the Player's statistical information for the play time in the season, for the player *petinga18*

in each position. The second pie chart, seen in Figure 5.13, represents the time played in each position and the percentage of play time in each position.

These pie charts are interactive, when hovering each section the information that is also written to the left of the pie chart will be shown. The reasoning behind representing the information in a written form besides the pie chart, is to more accurately specify what information is being represented. The information represented in both pie charts is relative to games played in the map Summoners Rift, and includes both ranked and non ranked playlists, and also the games played in any event mode are all added to a 6th variable, which represents any game type that does not have the conventional lane split of Top, Jungle, Middle, Bottom, and Support.

We created these two separate representations due to the fact that, the number and percentage of games played alone does not indicate the amount of time invested by a player into playing that position in the game. While having more games into a certain position, certainly indicates preference in playing that position, it does not, in most cases, indicate skill in playing said position. When paired with the time played in each position, we can then make better assumptions, by combining both statistics, on whether a certain player is proficient in playing a certain position or not. Of course, these statistics alone, cannot prove if a player is good at playing a certain position or not, but if paired with information on the games played in those positions, assumptions with greater reliability, could then be created into whether this player is more or less proficient in certain positions. Although currently, the only information we can acquire from these charts is the preferences of the player we are studying, when it comes to lane choice and lane playtime.

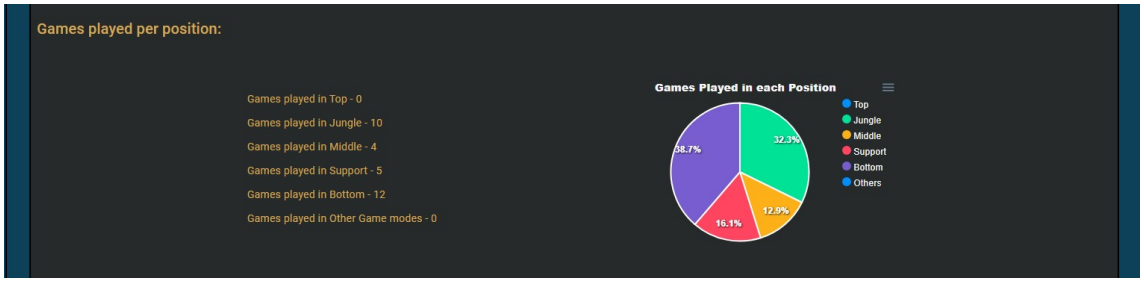


Figure 5.12: Pie chart representation of the Player's games divided by matches played in each position, for the player *petinga18*

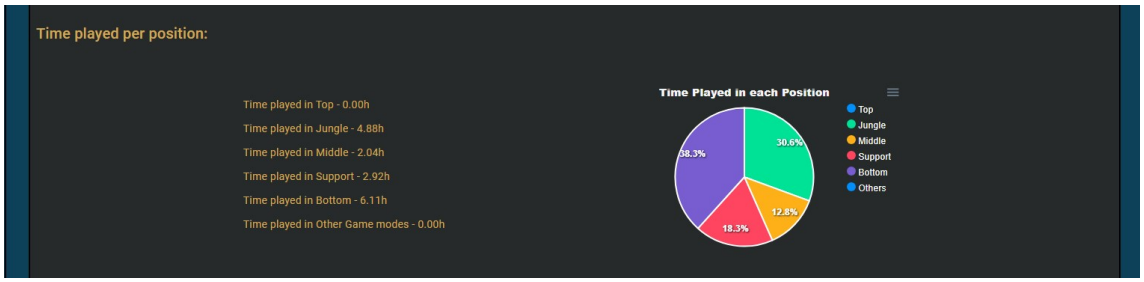


Figure 5.13: Pie chart representation of the Player's games divided by time played in each position, for the player *petinga18*

5.5.5 Line charts & Brush chart

Following the pie charts in the Player page, we designed the column charts (Figure 5.14 and Figure 5.15) and the brush chart (Figure 5.16). These two chart types are focused solely on playtime, although, each of them has their own focus on different playtime statistics. The main focus on these charts was to create a way to characterize players, and identify, possible groups of players they belong to. Through their playtime we could identify players and sort them into different groups, this is not something we currently do, but since the information is available, and can be gathered easily, this is something that could be done, or at least considered, in future projects or implementations that use the information shown on the prototype.

Now focusing on the charts themselves. First we present two column charts. These can be seen in Figure 5.14 and Figure 5.15, both with the same information but with different representations, which in their turn enable different conclusions. Following these, in Figure 5.16, we also present a Brush chart with the evolution of the total playtime of the player.

In Figure 5.14 we can see the total daily play times for the player for the duration of the Season. In this graph we represent the playtime for weekdays with blue, and the playtime for weekends with green. This type of graph can help if the player tends to be more active during the weekdays or the weekends. From this information we could infer if the player belongs to certain groups that tend to have more free time in their weekends.

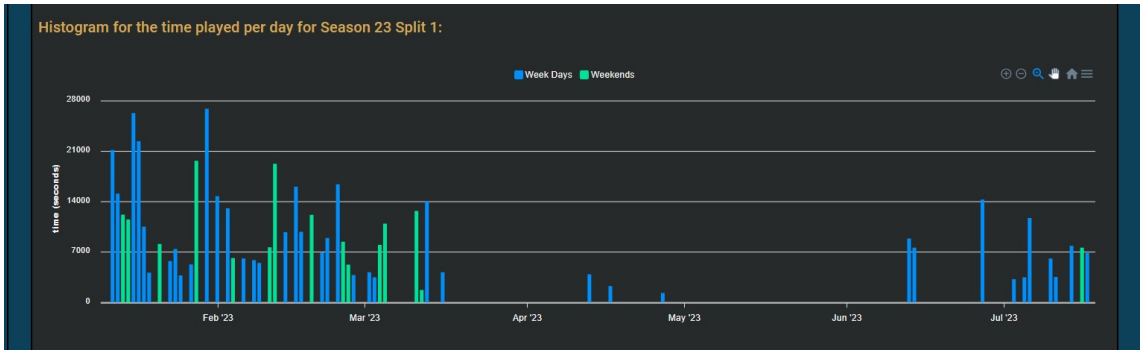


Figure 5.14: Column chart representation of the Player's playtime for each day with a split between week days and weekend days, for the player *AWT Love 36D*

In Figure 5.15 we can see the total daily playtime for the player in question, and we split the days into four periods, from 00:00 to 06:00, from 06:00 to 12:00, from 12:00 to 18:00, and from 18:00 to 24:00. This way, we can analyze in which periods the player is more active. Through this information, we can again try to deduce if this player belongs to a certain group of people that tend to have more free time to play, on certain time periods of the day.

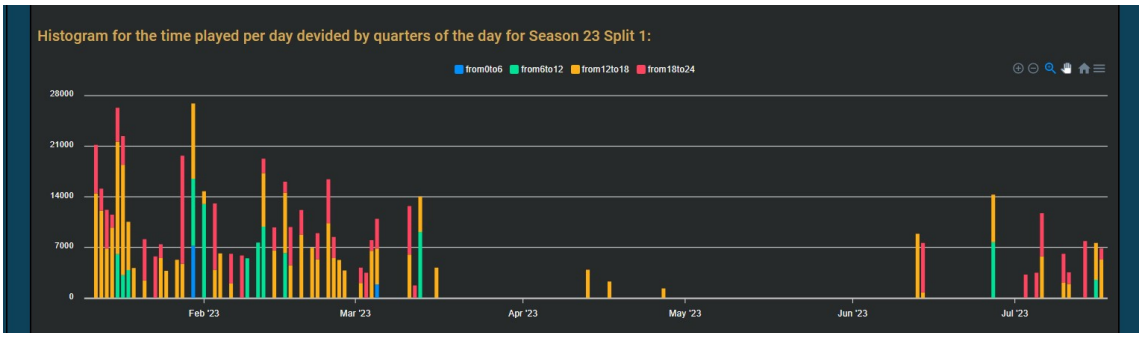


Figure 5.15: Column chart representation of the Player's playtime for each day split into quarters, for the player *AWT Love 36D*

Finally the chart in Figure 5.16 represents the playtime of the player throughout the season. This chart can be used to understand the evolution of playtime of the player. Through it we can understand the spikes of playtime the player has throughout the total time of the season, we can then understand when the player invested more time into the game by interpreting the chart. We can also see if the player is constantly playing the game at the same rate, or if the player invests his time into the game at specific periods of the season. For example we can see that the player invest most of his playtime at the beginning of the season. This could be justified do to the fact that the game changes at the beginning of the seasons therefore it might feel like a different experience and might pull players in with all the changes. The playtime eventually stagnates, and a smaller but noticeable spike of playtime can be seen at the end of the season, possibly justified due to a wish to either achieve a specific goal, or again a new mid season patch might have spiked the players' interest.

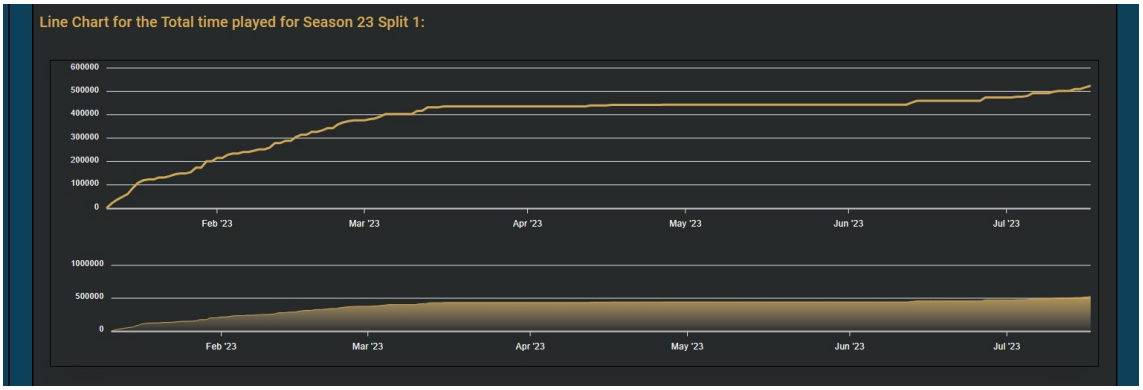


Figure 5.16: Brush chart representation of the Player's playtime, for the player *AWT Love 36D*

It is also important to state that, every chart shown in the previous images has some amount of

interact-ability, for which the most useful is the ability to zoom in and out in certain parts of the graph. This allows for a better understanding of the distribution for specific sections in the graphs. Most graphs can also be downloaded, with the exception of the brush chart.

5.5.6 Match list

Besides the Player statistical tab, we also have the Match List in a second tab of the player page. This tab was added in order to check the basic information for the games that are played by the player in question. For each match, we have the most important information of that match. We consider these to be the following, the champion played by the player along with their icon, the icon of the lane played, the kills assists and deaths for that match, along with the Kills Deaths Assists (KDA) that is calculated with Riot's formula:

$$KDA = \frac{Kills + Assists}{Deaths}$$

We also represent the game type of the match played in the JSON file (as described in Section 5.4.1), a button like representation of the end result of the match for the player, either game won or game loss, the match id for the match, along with the version the game was at when the match was played, the exact date that the game was created in, and finally the game duration in seconds. All of these statistics can be seen in Figure 5.17 for standard Summoner's Rift matches, and Figure 5.18 and Figure 5.19 for different gamemodes, for which the lane information is removed since it does not exist in their JSON representations.

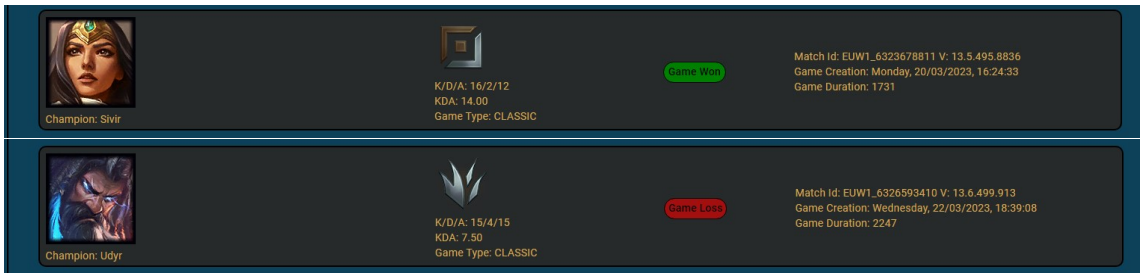


Figure 5.17: Two matches from the match list part of the player page, for the player *petinga18*, showing a game win and a game lost

Since we do not just look at the ranked matches of the players, this match list was important to add to the Player page. Through it, we can better understand the playing habits of the player. By analysing the match list we can understand how many matches the player plays in the regular game mode of LOL, or if the player prefers to play the less competitive game modes. From these less competitive game modes, some are available all the time, such as All Random All Middle (ARAM) 5.18, and some are limited and are only available at certain time periods, such as Ultra Rapid Fire (URF) 5.19. We do although, represent this information in the Pie charts, but we treat all non Summoner's Rift matches the same way. Therefore we consider that this representation is an important part of the match list. Besides the game type information, it is also important to

know, that if there was to be a better understanding of each match, a more in-depth focus would be required. But, since this was not in the main list of objectives, we deemed that, at least some information was vital to be presented.

This information, was selected in order to focus on different parts of the match information. When it comes to player performance and match outcome, we have the central part along with the left hand side of the representation. Then, on the right hand side of the representation, we have the more technical information of that match. This information, is important due to the vital details it gives us, such as the game Id. This Id can then be used to acquire the timeline, with the full match information and events that occur during the match. The reason for this information to be portrayed, although it adds nothing to the match portrayal, is that, it is vital information to infer if the player has a preference for any of the game types. Or, if a certain game patch was able to maintain the player for a longer period of time, than another.

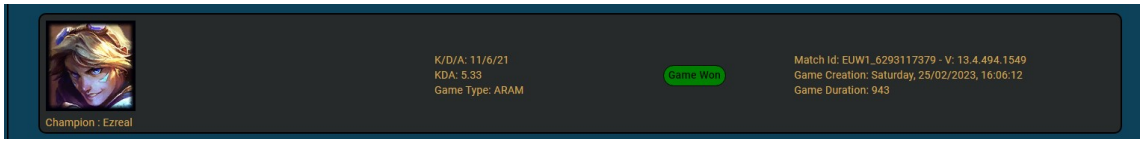


Figure 5.18: A match depicting an ARAM game, for the player *AWT Love 36D*

Figure 5.19: A match depicting an URF game, for the player *AWT Love 36D*

The one thing that is missing in the Match list view, is the ability to filter the match list. If this was to be added, along with a match count representation, then analysing all of this data, would become relatively easier. Again, since this was not one of the main points of focus, we ended up not giving the necessary focus to this feature. And therefore, ended up not implementing these additional features to the view.

5.6 Prototype Testing

In this section we will be going over the testing done for the prototype. In our case, we test the usability side of the prototype, through the use of the System Usability Scale (SUS) method [6] [49] [37], this was done in order to understand if the prototype was built in such a way that it could be easily learnt and used by the users.

5.6.1 User Testing

In order to test the prototype's usability, we started by carrying out a user study. This study was compromised of having users execute two tasks. These tasks consisted of having users interact

with the application to check some values and answer some questions. This was done in a way that the user had to actively traverse the application and interact with the full application. Following the tasks the users were presented with questions regarding the page they would traverse. We then, gathered the answers on the usability scale part of the form, and we calculated the SUS value. Using this value we could then use the SUS scale to understand if the prototype had a good usability value, or if in a future iteration, changes to the interface were also required [6] [49] [37].

The two tasks we had the users execute were the following:

- First we asked the user to go to the All players page and analyze the information that was presented to them. A couple of questions were asked about the information represented in the page. Following this, the task ended with the users attempting to add players to the database, this task can be seen in Appendix B.1.2.
- And for the second task we asked the users to return to the Home page and attempt to access a specific player's page (summoner *petinga18* from region *EUW*). While in the Player page they would once again be asked some questions that made them have to interact with the application and analyze the graphs presented to them or analyze the match list presented from that same player, this task can be seen in Appendix B.1.3.

We ended the form with the tasks by presenting the SUS method questionnaire in order for the users to give their opinion on our prototype's usability. The form for the SUS method can be seen in Appendix B.1.4.

5.6.2 Testing Results

In order to test the web application prototype, we gathered a population of 10 people for the testing process. We focused mostly on using users that previously played LOL or other MOBA style games. The population that tested the prototype was mostly a male population (70%), with an average age of 27.9 and a standard deviation of 2.02 (This can also be seen on Table 5.1).

Table 5.1: Characterization of the audience used for testing

Gender n (%)	Female	2	(20)
	Male	7	(70)
	Didn't disclose	1	(10)
Age (years)	Mean (SD)	27.9	(2.02)
	Min - Max	26 - 32	

From the testing process, we expected our prototype to have an average to good rating overall on the SUS scale. This is due to the fact that we attempted to make the prototype as simple and fluid as possible. Therefore, we expected the usability question on the SUS form to give us a higher average score than the ones were the users were asked if they would actively use our application regularly.

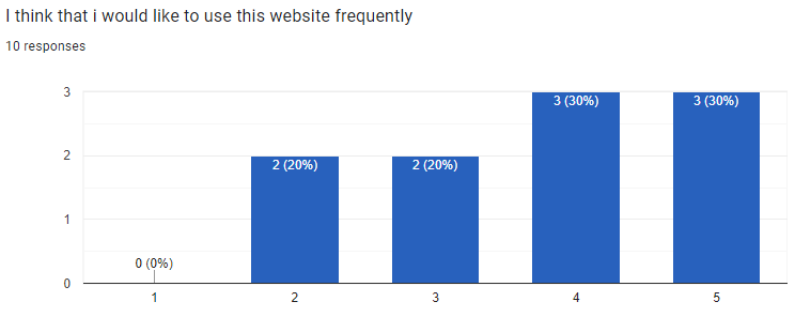


Figure 5.20: Question 1 in the SUS form questionnaire and the responses obtained

As can be seen in Figure 5.20, the average score in Question 1 is 3.7 we can infer that maybe the testing population used was not the target audience for the application, since the information presented to the users might be more useful to analysts or researchers than the average player. There are other possibilities, such as the fact that most of our testers used to play League of Legends, or other MOBA games, but currently they do not actively play said games.

Following the initial question, we split the remaining questions into two groups. The first group we defined as the questions that evaluate the ease of learning the prototype 5.23, 5.26, 5.28, 5.29, while the second group was defined as the questions that evaluate the ease of usage of our prototype 5.21, 5.22, 5.24, 5.25, 5.27. The results from both of these groups indicate that our tool is, at least for the users that tested, an easy tool to learn and also a tool that does not require technical help to be correctly utilized. In all of the following graphs we can see that the results either vary to either the highly agreeing side or the highly disagreeing side of the results. These were the expected results, although these could still be upgraded overall. The suggestions given by the testers can be seen in Chapter 6.2.

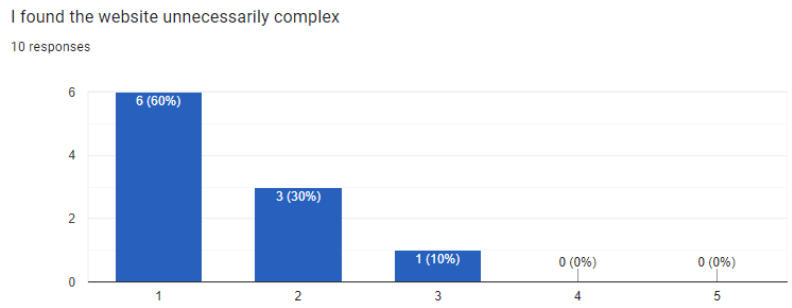


Figure 5.21: Question 2 in the SUS form questionnaire and the responses obtained

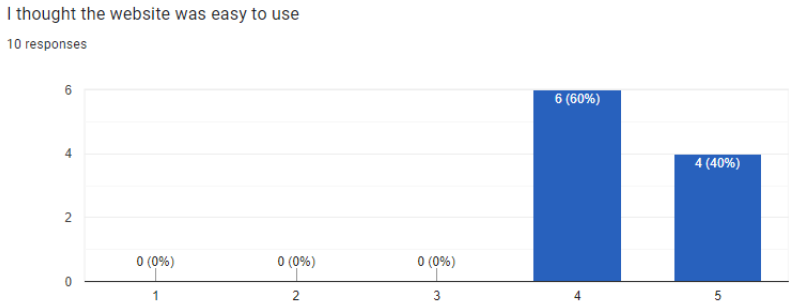


Figure 5.22: Question 3 in the SUS form questionnaire and the responses obtained

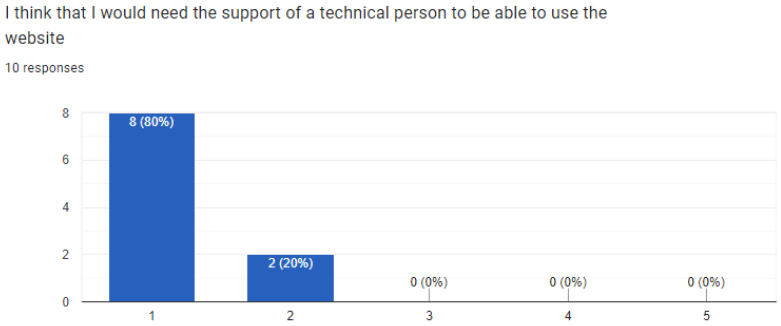


Figure 5.23: Question 4 in the SUS form questionnaire and the responses obtained

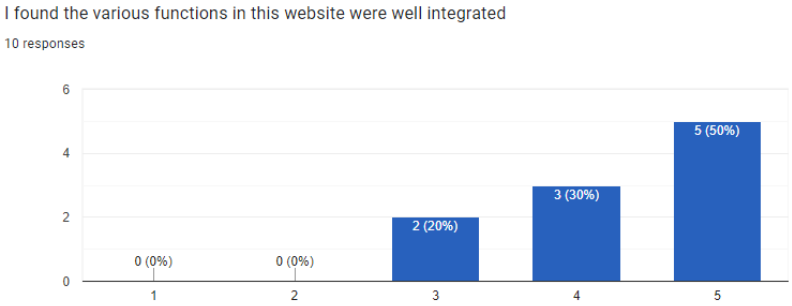


Figure 5.24: Question 5 in the SUS form questionnaire and the responses obtained

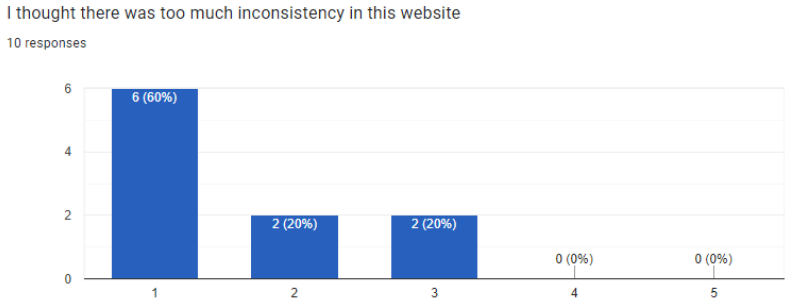


Figure 5.25: Question 6 in the SUS form questionnaire and the responses obtained

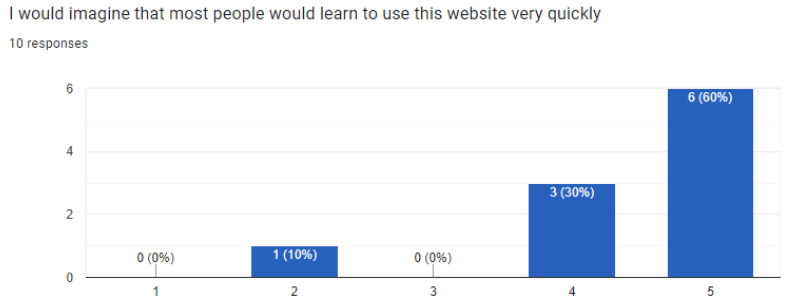


Figure 5.26: Question 7 in the SUS form questionnaire and the responses obtained

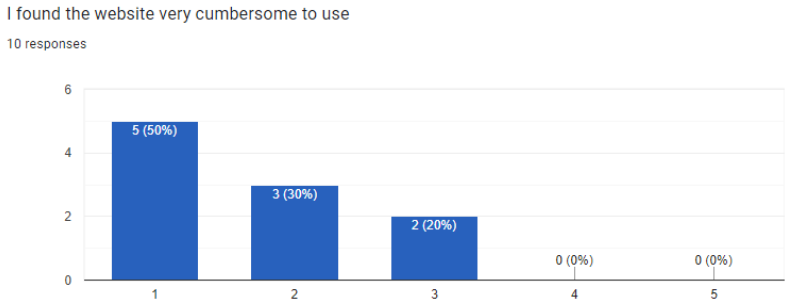


Figure 5.27: Question 8 in the SUS form questionnaire and the responses obtained

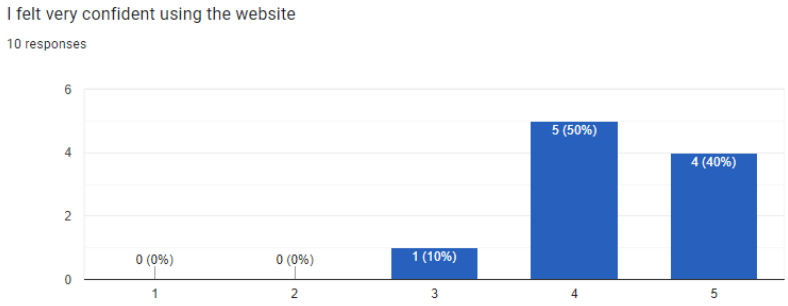


Figure 5.28: Question 9 in the SUS form questionnaire and the responses obtained

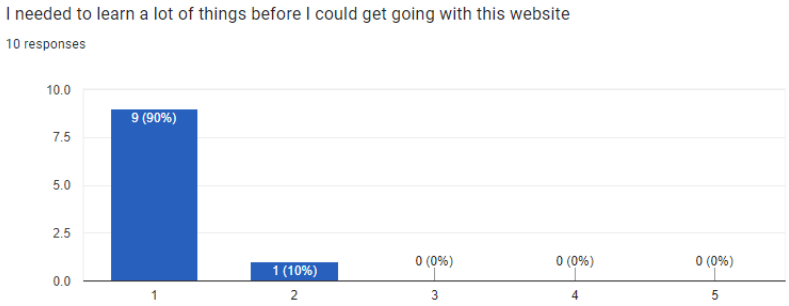


Figure 5.29: Question 10 in the SUS form questionnaire and the responses obtained

Once the tests were done, we took the results from the SUS forms and calculated the value for the SUS value of the prototype [6][37][53].

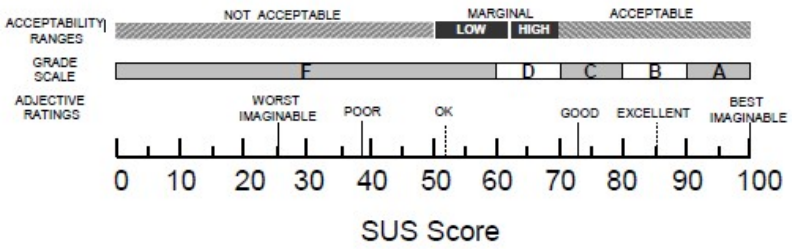


Figure 5.30: A comparison of the adjective ratings, acceptability scores, and school grading scales, in relation to the average SUS score [4]

The end result of the tests for our SUS tests had an average score of 85 this value is equivalent to a **B**, or an *excellent* having in mind the distribution of scores (as can be seen in Figure 5.30). The score obtained indicates that the usability of the prototype was rated to be acceptable, in future tests there should be a strive towards keeping this rating, and if possible we should attempt to upgrade our scoring since it could be higher in terms of usability of the prototype. We should also keep in mind the feedback obtained from some of the testers, since they also indicated possible solutions to issues they detected with the usability of the prototype. These changes will be mention in section 6.2.

5.7 Summary

Summarizing, we believe that the prototype, ended up filling the void that existed in previous applications developed by fellow colleagues. The focus, was to create an application that would be able to present the player statistics, and not his matches, in a way that highlighted the time invested by the player, in both a competitive and non-competitive environment of the game. Throughout this chapter we described the prototype, presented the architecture, explained the main visualizations built and finally we presented the testing that was done along with the results gathered. Besides this we also showed several points that were done although they were not initially seen as critical or even necessary, and when needed we also pointed out issues and possible additions that could have been done at the time of development. But most importantly, we presented the work done, and the intentions with the prototype we implemented. We showed the evolution whether that was through the architecture or the actual web app, and in the end we believe we achieved the main objectives.

Chapter 6

Conclusion & Future Work

In this chapter we will present our works conclusions in Section 6.1 and in Section 6.2 we go over the work that could be done in the future.

6.1 Conclusion

Through the research done, whether that was from the past thesis, the numerous papers found on player game data, or the several web applications that serve to assist and correct negative player habits, we can understand that the interest for these types of studies will continue to exist, which can be supported by the data found.

Initially in our study, we focused mainly into research and interpretation of other studies to better understand the possible advantages, and importance of our thesis. We found many studies that proved, that this type of information is just as important to categorize players as their way of playing is. Given that earlier studies did not focus on player data, our primary objective was to address that missing point of focus. Additionally it would also complement the previous research to enhance the understanding of playing habits and player intentions.

Given the objectives we set for ourselves, in all the iterations of our project, we feel like we were able to achieve most of what we set out to do. Although, there are things that can still be done in future iterations of our project, we believe that our main objectives were achieved. We were able to, explore the FPS genre of games and to understand the possibility of applying the techniques used in previous thesis to the genre. In our attempt, we found multiple possible games to use, but ended up choosing to use CS:GO. Unfortunately, our attempt ended up in a stump. Although we were not able, to apply previous knowledge to a new gametype, we still presented our findings, when it came to the exploration of the current services, which present player data, for FPS games. In this case, most of the services found show statistical data from matches, or are applications that are used to review the matches played. We then explored the APIs for the FPS genre games, were we focused on CS:GO and Valorant, and showed our discoveries when it came to the Steam API and the Riot API. We explored Valorant, in the hope of being able to find a workflow for data gathering for an FPS game, which also ended up not working, due to data access restriction imposed by the Riot API. In the end, since none of our attempts were successful, we presented the

work loop we found for CS:GO, which was able to obtain data. But, at the time of completion, we would still require to gather players, for which to gather information, and due to time implications this project iteration was dropped. We then, turned to LOL once again. Having this in mind, we attempted to explore and provide a deeper understanding of the ranked system of LOL. We believe we provide an in-depth summary of the most important concepts of the ranked ladder for LOL. Following this, we explored the Riot API and presented the restrictions that exist for API access, we presented these in detail in our thesis and we believe we summarized them when the necessity presented itself. Finally, we created a web application that could present player playtime data. We tested the application in team, although this is not the optimal way to do so, and in the end, we believe the product created was able to meet all the expectations required of it. These objectives were, in most cases, fully realized.

6.2 Future Work

When it comes to work that can be done in the future to enhance our project, there are several things that can be done in the prototype:

- The most important point is to better test the prototype. Since currently, the prototype is mostly built with the idea of no user failure. And the tests done on the current iteration of the prototype were just usability tests;
- Although the test results were favourable and quite positive, there were some points made by the testers, such as possible fixes and changes to upkeep the consistency of the prototype. Some testers suggested the addition of buttons on the players in the player list to access the players automatically from the All Players view. There was also the suggestion of adding return buttons on several pages to return to the Home page. These changes along with possible upgrades to the search bar with auto complete functionalities and the fix of crashes for searching for players that do not exist would greatly enhance the user experience;
- Other views for player information can be included into our prototype. This could require a restructuring of the prototype in general, both in it's core code and in it's UI. If a more specific analysis was made with players in mind, more information could be gathered and displayed, therefore enhancing the uses of our prototype;
- There can also be an attempt to acquire a Production key, through the processes given by Riot in their API website. If the outcome is positive, then there are many possible features that can be added to the prototype, such as, RSO login with Riot credentials, live gathering of data trough the Riot API, and other games from Riot could also be explored with the key, such as Valorant their take on a shooter, TFT their take on an autochess game, LOR their take on a card game, and possibly keep up with their future projects, these being 2XKO (formerly known has Project L)[16] their take on a fighting game, and the MMORPG game they have tease but not much is known about. The reasoning behind this is the availability that a higher

level API key enables for the games already launched by Riot, and the possibilities it enables for future projects to come;

- Given the work loop found to gather CS:GO match data, which technically would be the same for the current iteration of Counter Strike (Counter Strike 2). It could be considered, in order to apply the work loop to the game, and with a group of players gathered, attempt the construction of the dataset of gamedata for CS;
- Along the thesis, there are multiple points at which we mentioned certain features, that could in some cases solve issues, or be added to facilitate the interpretation of data, RSO Login, obtaining the Production key, implementing the player addition to the db and consider new visualizations and data to add to our views. All of these features, could be, in a future implementation considered. These, although not vital, are features that would enhance our product. Therefore, if the chance is provided, it is vital that they are implemented.

Bibliography

[1] Rafael Nuno Fragoso Afonso. Visualeague III: Visual analytics of multiple games. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2020.

[2] ApexCharts. Apexcharts. `https://apexcharts.com/`. Access: August 2023.

[3] ArenaNet. Games are art. we bring art to life. `https://www.arena.net/en`. Access: December 2021.

[4] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual SUS scores mean: adding an adjective rating scale. *J. Usability Studies*, 4(3):114–123, May 2009.

[5] Sebastian Block and Florian Haack. esports: a new industry. `https://www.proquest.com/conference-papers-proceedings/esports-new-industry/docview/2488515527/se-2`, 2021.

[6] John Brooke. *SUS – a quick and dirty usability scale*, pages 189–194. Taylor & Francis, January 1996.

[7] Nuno Narciso Carreiro. Técnicas de visualização para melhor desempenho em jogos online. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2016.

[8] Steam Charts. Steam charts - tracking what's played. `https://steamcharts.com/`, 2022. Access: March 2023.

[9] Valve Developer Community. Counter-strike: Global offensive access match history. `https://developer.valvesoftware.com/wiki/Counter-Strike:_Global_Offensive_Access_Match_History`. Access: March 2023.

[10] Valve Developer Community. Steam web API. `https://developer.valvesoftware.com/wiki/Steam_Web_API`. Access: March 2023.

[11] Sónia dos Santos Portugal Alves da Costa. Visual analytics of team strategies in online games. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2022.

[12] M.S. El-Nasr, T.H.D. Nguyen, A. Canossa, and A. Drachen. *Game Data Science*. Oxford University Press, 2021.

[13] Victor R. M. Feitosa, José G. R. Maia, Leonardo O. Moreira, and George A. M. Gomes. Gamevis: Game data visualization for the web. In *2015 14th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 70–79, 2015.

[14] OpenJS Foundation. Express. https://expressjs.com/. Access: August 2023.

[15] OpenJS Foundation. Nodejs. https://nodejs.org/en. Access: August 2023.

[16] Riot Games. 2xko 2v2 team-based fighting game - formerly known as project l. https://2xko.riotgames.com/en-us/. Access: May 2024.

[17] Riot Games. Riot developer portal. https://developer.riotgames.com/. Access: June 2024.

[18] Riot Games. Riot developer portal - API. https://developer.riotgames.com/apis. Access: June 2024.

[19] Riot Games. Riot developer portal - API keys. https://developer.riotgames.com/docs/portal#web-apis_api-keys. Access: June 2024.

[20] Riot Games. Riot developer portal - Data Dragon. https://developer.riotgames.com/docs/lol#data-dragon. Access: September 2023.

[21] Riot Games. Riot developer portal - RSO integration. https://developer.riotgames.com/docs/lol#rso-integration. Access: September 2023.

[22] Riot Games. Riot games - who are we? https://www.riotgames.com/en/who-we-are. Access: August 2023.

[23] Riot Games. Riot pls anniversary edition recap. https://www.riotgames.com/en/news/riot-pls-anniversary-edition-recap. Access: May 2023.

[24] Riot Games. The Valorant closed beta starts April 7. https://playvalorant.com/en-gb/news/announcements/the-valorant-closed-beta-starts-april-7/. Access: May 2023.

[25] Riot Games. Valorant closed beta ends may 28. https://playvalorant.com/en-us/news/announcements/valorant-closed-beta-ends-may-28/. Access: May 2023.

[26] Google. Angular. https://angular.io/. Access: August 2023.

[27] Google. Angular material. https://material.angular.io/. Access: August 2023.

[28] Christina Gough. Topic: Esports market. https://www.statista.com/topics/3121/esports-market/, October 2021.

[29] Jerome Heath, Isaac McIntyre, Mateusz Miter, and Eva Martinello. Start and end dates for all lol seasons. https://dotesports.com/league-of-legends/news/start-and-end-dates-for-all-league-of-legends-seasons. Access: July 2023.

[30] Erica Kleinman and Magy S El-Nasr. Using data to "git gud": A push for a player-centric approach to the use of data in esports, April 2021.

[31] Jeff Landa. Valorant - beginner's guide. https://playvalorant.com/en-us/news/announcements/beginners-guide/. Access: September 2023.

[32] Laserface. End of season 2023 - split 1. https://support-leagueoflegends.riotgames.com/hc/en-us/articles/17406448836243-End-of-Season-2023-Split-1. Access: August 2023.

[33] Laserface. MMR, rank, and LP. https://support-leagueoflegends.riotgames.com/hc/en-us/articles/4405781372051. Access: August 2023.

[34] Laserface. Placements, promotions, series, demotions, and decay. https://support-leagueoflegends.riotgames.com/hc/en-us/articles/4405783687443. Access: August 2023.

[35] Luxia Le. A complete history of valve: Founding, hits, and failures. https://history-computer.com/business/companies/a-complete-history-of-valve-founding-hits-and-failures/. Access: March 2023.

[36] Nikolas Martinos. Functional and non-functional requirements for a performance dashboard in CS:GO, July 2021.

[37] Ana Isabel Martins, Ana Filipa Rosa, and Ale Queirós. European portuguese validation of the system usability scale (SUS). *Procedia Computer Science*, 67:293–300, 2015. Proceedings of the 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion.

[38] Ben Medler, Michael John, and Jeff Lane. Data cracker: Developing a visual game analytic tool for analyzing online gameplay. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, page 2365–2374, New York, NY, USA, 2011. Association for Computing Machinery.

[39] Mickathia.8701. GW2 DPS reports. https://dps.report/. Access: December 2021.

[40] MongoDB. MongoDB. https://www.mongodb.com/. Access: August 2023.

[41] MongoDB. MongoDB compass. https://www.mongodb.com/products/tools/compass. Access: August 2023.

[42] Tiago Alexandre Orrico Moucho. Visualeague II - animated maps for performance analysis in games. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2018.

[43] The Esports Observer. https://archive.esportsobserver.com/the-esports-eco-system/, August 2017.

[44] Maryville University's online Bachelor of Science in Rawlings Sport Business Management. Evolution of the esports industry: An inside look. https://online.maryville.edu/blog/evolution-of-esports-industry/, September 2020.

[45] RiotAether - Paul Perscheid. /DEV: Ranked schedule changes. https://www.leagueoflegends.com/en-us/news/dev/dev-ranked-schedule-changes/. Access: September 2023.

[46] pnxenopoulos. awpy is a python library to parse, analyze and visualize counter-strike: Global offensive (CSGO) data. https://github.com/pnxenopoulos/awpy. Access: March 2023.

[47] Postman. Postman. https://www.postman.com/. Access: August 2023.

[48] Riot Auberaun - Chris Roberts and Riot Revenancer - Evan Humphreys. Whats's next for Ranked. https://www.leagueoflegends.com/en-us/news/game-updates/what-s-next-for-ranked/. Access: August 2023.

[49] Jeff Sauro. Measuring usability with the system usability scale (SUS), February 2011.

[50] Skittle Sniper. Master, grandmaster, and challenger: The apex tiers. https://support-leagueoflegends.riotgames.com/hc/en-us/articles/4405776545427-Master-Grandmaster-and-Challenger-The-Apex-Tiers. Access: August 2023.

[51] Steam. Register steam web API key. https://steamcommunity.com/dev/apikey. Access: March 2023.

[52] Steam. Steam support counter-strike: Global offensive game authentication codes. https://help.steampowered.com/en/wizard/HelpWithGameIssue/?appid=730&issueid=128. Access: March 2023.

[53] StuartAffect. System usability scale (SUS) score calculator, 2020. Third-party program to help the calculation of the SUS values and averages.

[54] Unknown. Arcdps: GW2 DPS meter (and general combat metrics tool). https://www.deltaconnected.com/arcdps/. Access: December 2021.

[55] Valve. At valve we make games, steam, and hardware. https://www.valvesoftware.com/en/about. Access: March 2023.

[56] Valve. Steam web API documentation. https://steamcommunity.com/dev. Access: March 2023.

[57] Valve. Steamworks. https://partner.steamgames.com/. Access: March 2023.

[58] ValvePython. Python package for interacting with CS:GO game coordinator. https://github.com/ValvePython/csgo. Access: March 2023.

[59] ValvePython. Python package for interacting with steam. https://github.com/ValvePython/steam. Access: March 2023.

[60] Pedro Miguel Almeida Vieira. Visualization of spatio-temporal information for personal performance analysis in games. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2017.

[61] G. Wallner and S. Kriglstein. Visualization-based analysis of gameplay data – a review of literature. *Entertainment Computing*, 4(3):143–155, 2013.

[62] Guenter Wallner. Automatic generation of battle maps from replay data. *Information Visualization*, 17(3):239–256, 2018.

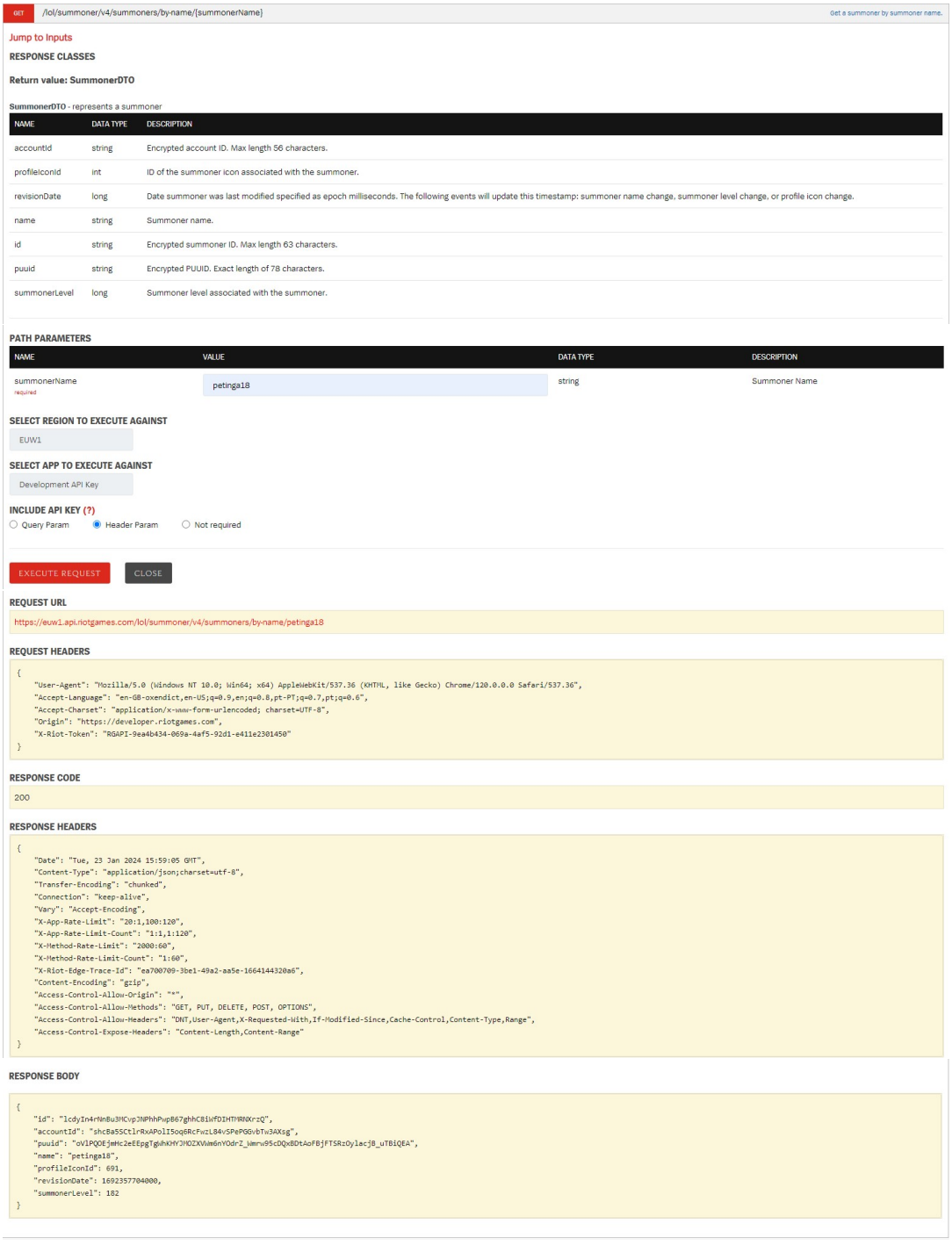
[63] Guenter Wallner and Simone Kriglstein. Visualizations for retrospective analysis of battles in team-based combat games: A user study. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '16, page 22–32, New York, NY, USA, 2016. Association for Computing Machinery.

[64] Günter Wallner, Nour Halabi, and Pejman Mirza-Babaei. Aggregated visualization of playtesting data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery.

[65] Zippia. Valve company history timeline. https://www.zippia.com/valve-careers-173484/history/. Access: March 2023.

Appendix A

Riot API Callback



GET /lol/summoner/v4/summoners/by-name/{summonerName} — Get a summoner by summoner name.

Jump to Inputs

RESPONSE CLASSES

Return value: SummonerDTO

SummonerDTO - represents a summoner

NAME	DATA TYPE	DESCRIPTION
accountId	string	Encrypted account ID. Max length 56 characters.
profileIconId	int	ID of the summoner icon associated with the summoner.
revisionDate	long	Date summoner was last modified specified as epoch milliseconds. The following events will update this timestamp: summoner name change, summoner level change, or profile icon change.
name	string	Summoner name.
id	string	Encrypted summoner ID. Max length 63 characters.
puuid	string	Encrypted PUUID. Exact length of 78 characters.
summonerLevel	long	Summoner level associated with the summoner.

PATH PARAMETERS

NAME	VALUE	DATA TYPE	DESCRIPTION
summonerName required	petinga18	string	Summoner Name

SELECT REGION TO EXECUTE AGAINST

EUW1

SELECT APP TO EXECUTE AGAINST

Development API Key

INCLUDE API KEY (?)

○ Query Param ◉ Header Param ○ Not required

EXECUTE REQUEST CLOSE

REQUEST URL

https://euw1.api.riotgames.com/lol/summoner/v4/summoners/by-name/petinga18

REQUEST HEADERS


```
{
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36",
    "Accept-Language": "en-GB-oxendict,en-US;q=0.9,en;q=0.8,pt-PT;q=0.7,pt;q=0.6",
    "Accept-Charset": "application/x-www-form-urlencoded; charset=UTF-8",
    "Origin": "https://developer.riotgames.com",
    "X-Riot-Token": "RGAPI-9ea4b434-069a-4af5-92d1-e411e2301450"
}
```


RESPONSE CODE

200

RESPONSE HEADERS


```
{
    "Date": "Tue, 23 Jan 2024 15:59:05 GMT",
    "Content-Type": "application/json;charset=utf-8",
    "Transfer-Encoding": "chunked",
    "Connection": "keep-alive",
    "Vary": "Accept-Encoding",
    "X-App-Rate-Limit": "20:1,100:120",
    "X-App-Rate-Limit-Count": "1:1,1:120",
    "X-Method-Rate-Limit": "2000:60",
    "X-Method-Rate-Limit-Count": "1:60",
    "X-Riot-Edge-Trace-Id": "ea700709-3be1-49a2-aa5e-1664144320a6",
    "Content-Encoding": "gzip",
    "Access-Control-Allow-Origin": "*",
    "Access-Control-Allow-Methods": "GET, PUT, DELETE, POST, OPTIONS",
    "Access-Control-Allow-Headers": "DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range",
    "Access-Control-Expose-Headers": "Content-Length,Content-Range"
}
```


RESPONSE BODY


```
{
    "id": "lcdyIn4rNnBu3MCvpJNPhhPwpB67ghhC8iWfDIHTMRNXrzQ",
    "accountId": "shcBa5SCtlrRxAPolI5oq6RcFwzL84vSPePGGvbTw3AXsg",
    "puuid": "oVlPQQEjmHc2eEEpgTgWhKHYJMOZXVWm6nYOdrZ_Wmrw95cDQxBDtAoFBjfTSRzOylacj8_uTBiQEA",
    "name": "petinga18",
    "profileIconId": 691,
    "revisionDate": 1692357704000,
    "summonerLevel": 182
}
```


Figure A.1: The request to Riot API to obtain player information for the player *petinga18*

Appendix B

SUS Questionnaire form

B.1 LoLChrono - Test Form

This form was created in order to analyze LoLChrono. LoLChrono is a League of Legends third party application that can be used to gather and analyze telemetry data from League of Legends players. Through this form we want to analyze the usability of our application and to do so we utilized SUS (System Usablity Scale) in order to test out our prototype's usability.

B.1.1 Identification

In order to identify the population that responded to our form, please answer the following questions:

Age:

_____

Gender Identity:

Male	
Female	
Prefer not to say	
Other	

B.1.2 Task 1 - All Player page

Please go to the player list section of the prototype and answer the following questions:

Q1:How many players are in the database?

4	
3	
5	
0	

Q2:What is the total number of matches played by the player "AWT Love 36D"?

163	
323	
167	
233	

Q3:Were you able to add the following player to the db?

- Name - *Petinga*;
- Region - *Europe Nordic East*.

Yes	
No	

B.1.3 Task 2 - Player page

Please go back to the home page, and access the following player:

- Name - petinga18;
- Region - Europe West.

Once you are located in this player's player page please answer the following questions:

Q1:What is the total playtime of this player in hours? (answer with the value in hours)

_____

Q2:What was the outcome for the most recent match of this player?

Victory	
Loss	

Q3:What was the day in which this player played the most time?

January 31st	
February 8th	
March 15th	
March 20th	

Q4:What was the percentage of playtime for Jungle lane for this player?

32.3%	
30.6%	
38.7%	
38.3%	

Q5:What was the champion used in the latest match for this player?

Sivir	
Ezreal	
Zeri	
Udyr	

Q6:How many matches did this player play in the Support role?

4	
5	
0	
10	

B.1.4 Question Usability Scale

In this section it is vital to read the questions with attention, please answer these questions honestly.

Q1: I think that I would like to use this website

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q2: I found the website unnecessarily complex

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q3: I thought the website was easy to use

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q4: I think that I would need the support of a technical person to be able to use the website

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q5: I found the various functions in this website were well integrated

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q6: I thought there was too much inconsistency in this website

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q7: I would imagine that most people would learn to use this website very quickly

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q8: I found the website very cumbersome to use

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q9: I felt very confident using the website

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Q10: I needed to learn a lot of things before I could get going with this website

	1	2	3	4	5	
Strongly disagree						Strongly Agree

Appendix C

SUS Questionnaire responses

C.1 LoLChrono Response Tables

C.1.1 Task 1 - All Player page task answers

Table C.1: Answers to the first task of the testing form

	Q1	Q2	Q3
1	4	323	Yes
2	4	323	Yes
3	4	323	Yes
4	4	323	Yes
5	4	323	Yes
6	4	323	Yes
7	4	323	Yes
8	4	323	Yes
9	4	323	No
10	4	323	Yes

C.1.2 Task 2 - Player page task answers

Table C.2: Answers to the second task of the testing form

	Q1	Q2	Q3	Q4	Q5	Q6
1	15.95h	Loss	March 20th	30.60 %	Zeri	5
2	15.95h	Loss	March 20th	30.60 %	Zeri	5
3	15.95	Loss	March 20th	30.60 %	Zeri	5
4	15.95	Loss	March 20th	30.60 %	Zeri	5
5	15.95	Loss	March 20th	30.60 %	Zeri	5
6	15.95	Loss	March 20th	30.60 %	Zeri	5
7	15	Loss	March 20th	30.60 %	Zeri	5
8	15.95	Loss	March 20th	30.60 %	Zeri	5
9	16h	Loss	March 20th	30.60 %	Zeri	5
10	15.95h	Loss	March 20th	30.60 %	Zeri	5

C.1.3 Answers Usability Scale

Table C.3: Ansers to the System Usability Scale form

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	5	1	5	1	5	1	5	1	4	1
2	3	1	5	1	3	3	5	1	5	1
3	3	2	4	2	4	1	4	1	4	2
4	4	2	4	2	4	2	5	3	4	1
5	5	1	5	1	5	1	2	1	4	1
6	2	1	4	1	5	1	5	2	5	1
7	2	1	4	1	4	2	4	3	4	1
8	4	2	4	1	5	1	5	2	5	1
9	4	3	4	1	3	3	4	2	3	1
10	5	1	5	1	5	1	5	1	5	1