INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Using Game Analytics to facilitate players information processing

ANDRÉ MIGUEL FERNANDES DO SACRAMENTO
Setembro de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Using Game Analytics to facilitate players' information processing

André Miguel Fernandes do Sacramento

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Software Engineering

Supervisor: Prof. Carlos Vaz de Carvalho

Evaluation Committee:
President:

Porto, September 28, 2025

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore the work presented in this document is original and authored by me, having not previously been used for any other end.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, September 28, 2025

Dedicatory

To my family and friends, who make life worth living.

Abstract

Game analytics have become an increasingly relevant tool in understanding player behavior, identifying performance patterns, and supporting informed decision-making in competitive environments. In complex games such as League of Legends, the volume and granularity of available data frequently overwhelm players, and statistical information is often poorly communicated or outright omitted, impacting player experience, limiting their ability to process statistical data and transform raw statistics into actionable insights.

This thesis addresses the research question of how to extract and analyze game information to positively impact player performance and experience. Using Design Science Research Methodology, a contextualized game analytics framework was developed that transforms raw statistical data into meaningful, actionable feedback. The solution consists of a web-based application that retrieves match information through official APIs, applies statistical thresholds for comparative analysis, and generates personalized conclusions highlighting player strengths and improvement areas.

The system was implemented using modern web technologies and evaluated through a comprehensive user survey with 71 participants. Results demonstrate significant user acceptance, with 87% of participants reporting prior use of analytics tools, 80% indicating they modified their gameplay based on feedback, and 86.2% perceiving performance improvements from game analytics usage. The findings validate that properly contextualized game analytics can effectively facilitate player information processing and enhance gaming experience.

This research contributes a validated framework for transforming complex gaming statistics into accessible insights, demonstrating how software engineering approaches can address information processing challenges in competitive gaming environments. The thesis acknowledges limitations including demographic bias toward experienced players and suggests future research directions in cross-game validation and longitudinal impact assessment.

Keywords: Game Analytics, Information Processing, Competitive Games, Player Performance, Performance Feedback

Resumo

A Análise de Jogos tornou-se uma ferramenta cada vez mais relevante para compreender o comportamento de jogadores, identificar padrões de desempenho, e apoiar tomadas de decisão informadas em ambientes competitivos. Em jogos complexos como League of Legends, o volume e granularidade dos dados disponíveis sobrecarrega frequentemente os jogadores, e informação estatística é por muitas vezes mal comunicada ou simplesmente omitida, impactando a experiência do jogador, limitando a sua habilidade de processar dados estatísticos e transformar dados em bruto em iniciativa própria.

Esta tesea aborda a questão de investigação de como extrair e analisar informação de jogos para impactar positivamente a performance e experiência de jogadores. Usando *Design Science Research Methodology*, foi desenvolvida uma estrutura contextualizada de análise de jogos que transforma dados estatístico em bruto em feedback significativo. A solução consiste numa aplicação web que obtém informação de jogos através de APIs oficiais, aplica limites estatísticos para análise comparativa e gera conclusões personalizadas, destacando pontos fortes e áreas de melhoria para os jogadores.

O sistema foi implementado utilizando tecnologias web modernas e avaliado através de um inquérito abrangente com 71 participantes. Os resultados demonstram uma aceitação significativa por parte dos utilizadores, com 87% dos participantes a reportar uso prévio de ferramentas de análise, 80% a indicar que modificaram o seu estilo de jogo com base no feedback, e 86,2% a admitir a perceção de melhorias de desempenho através do uso de análise de jogos. Os resultados validam que a análise de jogos adequadamente contextualizada pode eficazmente facilitar o processamento de informação por parte dos jogadores e melhorar a sua experiência.

Esta investigação contribui com uma estrutura validada para transformar estatísticas complexas de jogos em conhecimentos acessíveis, demonstrando como as abordagens de engenharia de software podem abordar desafios de processamento de informação em ambientes de jogos competitivos. A tese reconhece limitações incluindo enviesamento demográfico em direção a jogadores experientes e sugere ações futuras de investigação em validação com outros jogos e avaliação de impacto longitudinal.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter aims to introduce the focus of the thesis, namely the usage of game analytics to facilitate players' data processing. It also contextualizes it, in today's landscape of games, and exposes its current problems, in regards to the difference between information that is readily available to players and the one that is hidden and only accessible through third party sources.

The goals of the work are established (like studying the environment surrounding data analytics in games, identifying flaws and developing an adequate solution), as well as the challenges it faces and constraints it must comply with and work around, mainly limits regarding the available data collection, which is then complemented by the chosen approach to tackle all of these aspects. Ethical considerations are also instigated, mainly as a way to assure users that the tasks carried out are compliant with relevant codes of conduct.

1.1 Context

The video game industry has seen constantly increasing growth in recent times, in popularity, investment, revenue, and number of games, among others. The online variant has been particularly appealing to players worldwide and, among these, multiplayer games (especially in the competitive variant) show the best prospects in the aforementioned statistics, with no sign of slowing down.

In competitive multiplayer games, statistical performance data serves as a critical but often underutilized form of communication between the game and players. While such games readily provide obvious information such as tutorial instructions and immediate feedback like scores or health bars, the deeper statistical insights about player performance, such as efficiency metrics, comparative rankings, and behavioral patterns, remain largely hidden or poorly presented. These game analytics represent a wealth of information that could help players understand their skill progression, identify improvement areas, and make informed decisions about their gameplay strategies. However, most competitive multiplayer games present raw statistical data in formats that are difficult for players to interpret meaningfully, or place this kind of information in obscure, poorly accessible mediums. The accurate collection, presentation and contextualization of statistical information is therefore crucial for enabling players to process complex performance data and translate it into actionable insights for skill development and enhanced gaming experience.

1.2 Problem

In competitive multiplayer games, players frequently require detailed statistical information to assess their performance, understand gameplay trends, identify improvement areas, and make strategic decisions. However, the statistical data and analytical tools provided within games are often insufficient to meet players' analytical needs. Raw match statistics, when available at all, are typically presented without context or comparative analysis, making it difficult for players to interpret their significance or translate them into actionable insights. Metrics are regularly omitted in the game's interface, preventing players from accessing, interpreting, comparing, and acting upon crucial performance data.

This poor presentation of statistical information creates several challenges. Players struggle to identify specific areas for improvement, as raw data fails to highlight meaningful patterns or benchmarks. Without comparative context, players cannot assess whether their performance metrics are above or below expected levels for their skill tier. The absence of actionable insights means that even when players can access their statistical data, they often cannot translate these numbers into concrete steps for improvement.

Consequently, players either ignore available statistical information entirely due to its complexity or absence, or spend significant time manually researching and interpreting data across multiple sources to gain meaningful insights. This situation leaves many players unable to leverage the wealth of performance data that games generate, limiting their potential for skill development and strategic improvement.

1.3 Objectives

Through this document and the work set out by it, the primary objective of this research is to investigate how game analytics can be designed, analyzed, and presented to facilitate player information processing in competitive multiplayer games. This overarching goal is achieved through the following specific research objectives:

1. **Analyze the current state of game analytics and identify information processing challenges in competitive gaming.**

   This objective involves conducting a comprehensive literature review of existing data analytics and game analytics approaches, tools, and methodologies. The research will examine how current solutions present statistical data to players and identify gaps in information accessibility and contextualization. This analysis will establish the theoretical foundation for understanding how players process complex gaming information and the barriers they face in translating raw data into actionable insights.

2. **Develop a framework for contextualized game analytics that enhances player information processing.**

   Based on the literature review, this objective focuses on designing a theoretical and practical framework that transforms raw statistical data into meaningful, contextualized insights. The framework will incorporate statistical thresholds, comparative analysis, and user-friendly presentation methods to bridge the gap between complex data and player comprehension. This includes defining the architectural principles, data processing methodologies, and user interface considerations necessary for effective information conveyance.

3. **Implement and validate the proposed framework through a League of Legends analytics tool.**

   This objective involves the practical implementation of the designed framework using League of Legends as a representative case study of competitive multiplayer games. The implementation will demonstrate the feasibility of the proposed approach and serve as a concrete example of how game analytics can be designed to facilitate information processing. The choice of League of Legends provides access to a vast, comprehensive API and a large user base for validation purposes.

4. **Evaluate the effectiveness of the analytics tool in facilitating player information processing and performance awareness.**

   The final objective focuses on empirically assessing whether the developed tool successfully addresses the identified information processing challenges. Through user surveys and feedback analysis, this evaluation will measure user perception of the tool's effectiveness, its impact on gameplay understanding, and its ability to provide actionable insights. This assessment will validate the proposed approach and help to understand the relationship between game analytics design and player information processing.

1.4 Ethical Considerations

This research adheres to established ethical principles in computing and data handling. All game data utilized is publicly accessible through official APIs provided by Riot Games and complies with their terms of service, privacy policy, and ethical guidelines. Since this data is already made publicly available by the game provider under their established ethical framework, its use for analytics purposes falls within acceptable ethical boundaries.

The analytics tool will process publicly available match statistics and performance data that players have implicitly consented to share through their participation in online play. No private personal information (such as contact details or real-world identifiers) will be collected, stored, or processed.

For the evaluation survey, participants provided informed consent and were assured of data anonymity. Survey participation was entirely voluntary with the right to withdraw at any time.

Furthermore, the project follows the Association of Computing Machinery's Code of Ethics and Professional Conduct (*Association for Computing Machinery* 2023) throughout its course, ensuring integrity and responsibility throughout the research process.

1.5 Approach

This research addresses the problem of inadequate statistical information processing in competitive multiplayer games through a systematic, multi-phase approach. The work follows a design science paradigm, progressing from theoretical understanding to practical implementation and empirical validation.

The approach begins with establishing a comprehensive understanding of existing game analytics solutions and their limitations through literature review. This foundation informs the design of a framework that transforms raw statistical data into contextualized, actionable insights for players.

The framework is then implemented and validated using League of Legends as a representative case study, demonstrating practical feasibility and real-world applicability. Finally, the solution's effectiveness is assessed through user evaluation, measuring its impact on player information processing and performance awareness.

This sequential approach ensures that the developed solution is both theoretically grounded and practically validated, contributing to the understanding of how game analytics can facilitate player information processing.

1.6 Research Methodology

This research employs the Design Science Research Methodology (DSRM), a methodology for developing and evaluating IT artifacts that address identified problems (Peffers et al. 2007). The methodology consists of six sequential activities that guide the research process from problem identification to communication of results:

Problem Identification: The problem is identified and defined, demonstrating its importance and justifying the value of a solution. This is set forth in the subchapter 1.2.

Objectives of Solution: Clear research objectives are established to guide the development of a solution based on the previous problem definition and knowledge of what is possible, as observed in subchapter 1.3.

Design and Development: The artifact' design, architecture, functionality, and implementation are approached and detailed. This can be found in chapters 3, 4 and 5.

Demonstration: The artifact's use is demonstrated to solve one or more instances of the problem, showing its feasibility and effectiveness. This is addressed in chapters 4 and 5.

Evaluation: The artifact is rigorously evaluated to determine how well it addresses the problem, typically through empirical methods such as surveys, experiments, or case studies, as shown in chapter 5.

Communication: Results are communicated to appropriate audiences through academic publications, presentations, or other scholarly venues.

1.7 Structure

The dissertation is structured in adequately divided chapters, characterized as follows:

- **Chapter 1 - Introduction**

  This introductory chapter contextualizes the research topic, presents the problem of inadequate statistical information processing in competitive games, enumerates the research objectives, outlines the methodological approach, addresses ethical considerations, and provides the document structure.

- **Chapter 2 - State of the Art**

  This chapter presents a comprehensive literature review of data analytics and game analytics, and establishes the theoretical foundation for the proposed solution.

- **Chapter 3 - Conception and Design**

This chapter details the solution conception process, requirements engineering, system architecture design, providing the theoretical and technical foundation for the implementation.

- **Chapter 4 - Implementation**

  This chapter describes the practical development of the game analytics tool, including technology choices, system architecture implementation, API integration, and development methodologies employed.

- **Chapter 5 - Evaluation**

  This chapter presents the empirical evaluation of the developed solution through user surveys, analyzing the tool's effectiveness in facilitating player information processing and its perceived impact on gameplay understanding.

- **Chapter 6 - Conclusions**

  This final chapter synthesizes the research findings, discusses the contributions and limitations of the work, and identifies directions for future research and development.

Chapter 2

State of the Art

This chapter aims to go over game analytics, creating a comprehension of the concept and expanding on it. It shows off very basic concepts of data analytics as a starting point, then proceeds to present the current landscape of game analytics, with an overview of commonly used techniques. The main goal is to produce an insightful knowledge foundation capable of aiding and justifying decisions further down the line.

2.1 Data Analytics

Weird as it may sound, the designation "data mining" actually precedes data analytics. It was originally defined as extracting knowledge from data (Fayyad 1996) and, in the context of human interaction, this knowledge can be interpreted as "interesting patterns that are generally valid, novel, useful, and understandable to humans" (Runkler 2020). This definition ties particularly well into the topic of this thesis, where information not effortlessly available through original sources is gathered and organized in ways that allow for the definition of meanings and inferences by the user.

Following the study of data mining, the term data analytics surfaced, and though many definitions have been constructed and validated through the years, the one adopted for this thesis is very akin to data mining, but applied to a broader branch of knowledge and usually led by computer analysis. A science that analyzes raw and rudimentary data with the objective of extracting knowledge and deriving arrangements and deductions from them (Moreira, Carvalho, and Horvath 2018, Fayyad 1996). Information is drawn from a particular source, to be inspected, transformed and modeled as seen fit, in an attempt to attain conclusions and support hypothesis.

2.1.1 Data Collection

The first step in any practical process concerning Data Analytics is the assembly of information to then organize, scrutinize, model and interpret. It's essential to the very concept of analysis, as this needs a subject, and without data no conclusions can be reached. For instance, in the business side of things, it is often painted as a means to improve customer experience, while helping guide decisions in companies, promoting growth and extending its influence (*What is Data Collection? Why is it Important for Your Business?* 2023).

The user benefits of this practice are easily observable in areas like healthcare. Patient data is used to drive efforts in research, communicate status and speed up processes, reduce manufacture and purchase costs, prevent outbreak spread and keep people informed, etc (*Importance of data collection in healthcare* 2023, *The Importance and Benefits of Data*

Collection in Healthcare 2023). During the recent epidemic of SARS-CoV-2, known as COVID-19, it was not only used to inform individuals worldwide (through web apps like `https://coronavirus.jhu.edu/map.html`, a map that provided live updates of the world status), but also helped rapidly develop and test a vaccine. It actively helps in saving lives in this field, and in many others it can be used to facilitate people's day-to-day.

However. it's also one of the main topics of discussion of the present, with ethical implications driving the conversation, specifically in the world overrun by technology we live in, with the recent public disclosure of the full lengths that web cookies are capable of reaching. Web cookies is the name attributed to the data harvesting performed by web apps, initially as a means to maintain certain information locally in favor of said apps' performance, purely beneficial to both users and service providers (Cahn et al. 2016). However, over time, this mechanism has been adapted to collect more and more knowledge of individuals and their tendencies (namely by Data Brokers), which is then sold to companies that abuse it to set apart users and divide them into groups, to then aggressively target these groups with ads and product placements, as well as influence the contents they see and the opinions that surround them, molding their view and swaying their own opinions. One of the most discussed instances of this is the Facebook - Cambridge Analytica Data Scandal (Rehman 2019), where data harvesting was misused to alter the political perception of the USA's voters and benefit certain candidates.

2.1.2 Data Processing

Having data in our hands is not sufficient to begin a proper analysis, as it can be duplicated, insecure, outdated, or faulty in other ways. Even with such flaws bridged, it may not yet be ready for examination, as its state and/or format may not be adequate for the intended procedure. As such, it must be processed before an actual exploration takes place.

2.1.2.1 Data Cleaning

The detection and repairing of faulty data (labeled "dirty data" in the area) is a crucial step in assuring that analysis will occur using suitable, useful, accurate information. Failure in doing so will result in imprecise statistics, wrong conclusions and untrustworthy decision-making. Accordingly, data cleaning should be a prerequisite in an Analytics process.

It consists of two major steps: error detection and error repairing (Chu et al. 2016). The first of these aims to identify anomalies, by defining Error Type ("What to detect?", can go from data itself to the rules of data collection used to reach it), Means ("How to detect?", either human-handled or automated) and the Business Intelligence Layer ("Where to Detect?", the stage at which the fix must be applied).

2.1.2.2 Data Transformation

With (assumably) clean data, as aforementioned, we should now focus on preparing it for assimilation, molding it to our needs in an attempt to reduce the necessary amount of work that is to come. This process is called Data Transformation.

Transformation is composing the existing data in a way that facilitates statistical interpretation and eases comparison to other sources (Manikandan 2010). It can be as simple as deriving a new value from applying rules/functions to existing values, aggregating data, or simplifying attributes like dates to an atomic level. It's widely used with the intention of

normalizing information (attaining a normal distribution) (DeCoster 2001), whose benefits are evident when pinpointing trends, and can also aid in simulating missing data.

2.1.3 Types of Analytics

After having the data processed, it can now be analysed in particular contexts, as seen appropriate, and used to draw conclusions and drive decision-making. The mentioned contexts are as follows.

2.1.3.1 Descriptive Analytics

The subject of Descriptive Analytics dwells on historical data to provide both quantitative and qualitative information; it interacts with past data to identify flaws and merits and draw conclusions, further using these to conduct necessary decision-making and reviews (Gudivada 2017).

In the area of Software Engineering, we have had contact with the model FURPS, an acronym for Functionality, Usability, Reliability, Performance and Supportability (Samadhiya, Wang, and Chen 2010). All of these quality factors are used to determine if a system/service is adequate and up to standard, and are all examples of Descriptive Analytics, through which the values of the metrics are avowed (Gudivada 2017).

2.1.3.2 Diagnostic Analytics

Also pertaining to data set in the past, and as the name suggests, Diagnostic Analytics focus on identifying root causes of specific previous occurrences, aiming to uncover why they happened (Gudivada 2017). It is overall seen as a more advanced kind of analytics, and is particularly useful in analysing production inefficiencies and when applied to subjects like disaster recovery (for instance, in the aviation industry).

2.1.3.3 Predictive Analytics

While still based on past data, Predictive Analytics relates to future events, and is set up on using historical data applied to statistical models in order to generate forecasts and predict trends (Gudivada 2017). It is very apparent in market studies for projected product success and tendency identification (for example, for publicity purposes).

It ultimately relies on the definition of predictive models through careful collection, structuring and massaging of data, combined with probability theory and regression analysis, models that result in predictions, aimed to guide decision-making whilst being under continuous monitoring (Kumar and Garg 2018).

2.1.3.4 Prescriptive Analytics

Unavoidably linked to its Diagnostics counterpart (not just in the allusion to Medicine), Prescriptive Analytics intend to explore an identified problem, originally revealed by Diagnostics, and aid in its resolution. It simulates resolution scenarios and determines an optimal solution, using the number of good and bad outcomes as criteria (Gudivada 2017). As such, an adequate answer for an issue is one where its simulation not only reveals a high amount of positive results, but does so while maintaining a low number of negative ones.

2.2 Game Analytics

With a basis now set for the understanding of Data Analytics, the procedures behind them and the motivations of their usage, we can move to exposing how such practices can be applied to the Games industry, the changes in philosophy and objectives, the main goals it aims to achieve and how it is being used in today's world.

2.2.1 Game Analytics: Data Analytics applied to games

In the dawning of games, making one was as "simple" as having an idea, executing that idea, and making it available to the public. The world simple is disingenuous for the actual work that entailed putting together an enjoyable and captivating game, but it is accurate when considering the process that goes into it nowadays. Gaming has evolved to become one of the biggest entertainment industries in the world with an astronomical accumulated market value(Research 2023), and as a consequence their development has evolved from an imprecise rule-free environment to a standardized, well-structured, carefully thought out process with massive companies like Nintendo, Microsoft and Sony ruling the market.

Something that arose with this growth spurt is the necessity of monitoring players and their habits, games and their trends, markets and their appeal, genres and their popularity, data and its significance. Analytics went from being non-existent, to somewhat relevant, to omnipresent, offering insights into target audience, market potential, balancing prospects, profit and success mensuration, etc. Their reach is far too great to go over in just a book, let alone this single chapter.

Game Analytics base themselves on Telemetry, data attained over a distance, comprising any transmitted signal (El-Nasr, Drachen, and Canossa 2016). While it is very akin to the general sense of information, it mainly intends to encompass digital data and is usually client server based, given the interest in collecting specifics on many players. There are usually servers dedicated solely to receiving telemetry, which means the games are connected to the servers and are tasked with sending all information. In parallel to what was introduced before, this information is then processed, stored, and analysed, generating results such as detailed statistics and predictions.

2.2.2 Trends in Game Analytics

As was previously introduced, Game Analytics have grown to enormous proportions. They can refer to topics like Player Analytics, Game Development Analytics, Game Publishing Analytics, Distribution Channel analytics, Game prediction analytics, Game data visualization, etc (Su, Backlund, and Engström 2021). These can all then be further segmented down into more specialized terms and circumstances. However, for the context of this thesis, we will focus solely on Player Analytics, with a particular focus on Competitive Multiplayer Games.

2.2.2.1 Game Data Mining

The term Data Mining was previously referred to as extracting knowledge from data. In the context of games, this concept takes a grander scale. Commonly misattributed to only the gaming community itself and the fan acquisition of data present in game files through exploration (sometimes illegally), it comprises methods used to scale data collection and that intend to act upon large datasets from the get-go (El-Nasr, Drachen, and Canossa 2016). It can explore answers to questions like "What are the geographical patterns behind

a given game?", "How much do people play games?", "How can I predict a player's next action?", "What flaws in game design can I fix?", etc.

Game Data Mining applies to bigger scales and contexts, and follows a standardized procedure in data acquisition. A Knowledge Discovery Process is put in motion, starting with research, passing through understanding, preparing and modeling data, to finally evaluating it and deploying results. However, large amounts of outcomes are generated, and not only must one know how to navigate through it, there should also be ways to discern between valuable information and waste to be discarded, referred to as measures.

2.2.2.2 Data Clustering

With large amounts of data like the aforementioned one, and in large scale analysis like player behavior, interpreting individual results is a gargantuan task whose value can't be justified due to the time it would take to complete, even for a computer. As such, clustering can be applied, as seen on figure 2.1, a method which (coupled with classification) reduces the dimension of a dataset and summarizes it to the its most important features, allowing faster processing and interpretation.

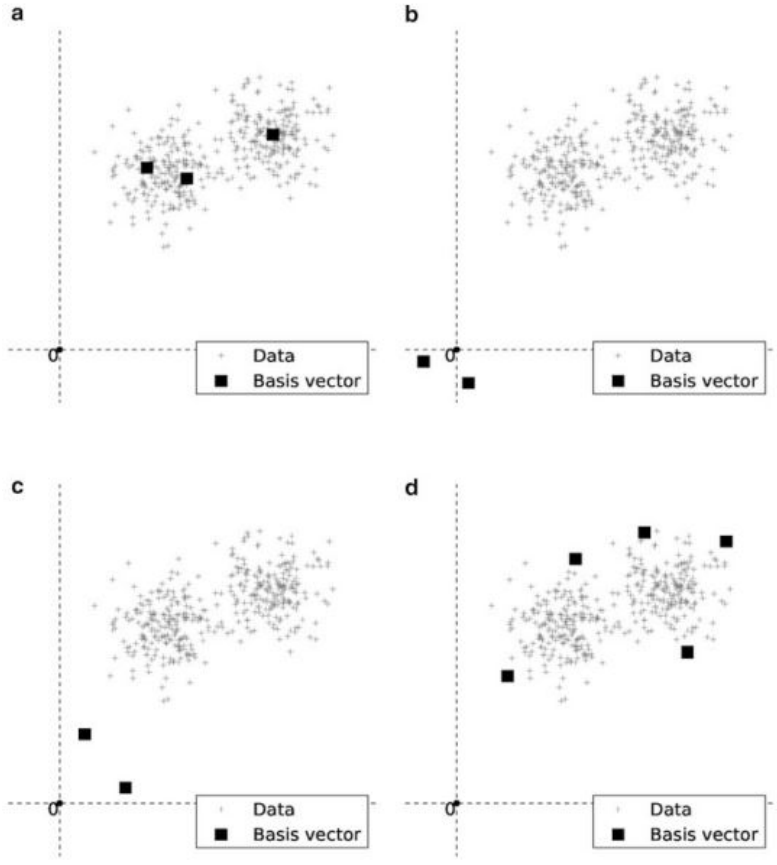


Figure 2.1: Example of Clustering and different factorization methods (El-Nasr, Drachen, and Canossa 2016)

2.2.2.3 Player Behavior

One of the most used measurements and characterizations of players used is the modeling of player behaviors. It is a crucial but complicated duty, where certain bits of data are assigned to specific behaviors, interpreted as the response a player gave to certain aspects present in gameplay or surrounding it (Su, Backlund, and Engström 2021). Monitoring every single possibility of response is not an option due to the sheer nature of options available, but not tracking at all is a severe loss for developers and publishers, and as such exploratory and evolutionary analyses are recommended, where player types are defined to reach initial criteria in relevant variables, to then develop from there and continue grasping better data (El-Nasr, Drachen, and Canossa 2016).

2.2.2.4 Player Segmentation

Having data on players is not enough of a basis to draw conclusions from and, as we saw before, segmenting players into groups can be widely beneficial to the developers when attempting to derive data from statistics. Players' characteristics, aspirations, requirements and capabilities can be asserted to arrive at specific player groups, which will contextualize information, discussions, statistics and consequent decision-making in meaningful and justified ways. This segmentation is based upon player region, behavior, peripheral information, time spent, money spent, etc (Su, Backlund, and Engström 2021).

2.2.2.5 Popular Types of Reports

Below are generated reports that gained popularity not only among developers (and analytics staff) due to their statistical importance, but also among players, with their capability of satiating the curiosity of those often "in darkness" when it comes to in depth data (Su, Backlund, and Engström 2021).

- Player Session Information

  This can include various details like: time spent per session; Win Ratio across sessions; comparison data between aspects like day of the week and time of day; time spent actively playing versus time spent waiting for a match to be found, etc.

- Active Users

  One of the most sought after outcomes of analytics that drive both internal and external discussions on the success of a game are depictions of the number of active users. These can range from days to years, and are some of the most used basis for online discussions on how well a game is received and how its popularity is set to evolve.

Figure 2.2: Chart of Active Players for CS2, provided by SteamDB (*CS2 Active Players* 2024)

- New Users

  The amount of new users a game has accrued over a set period of time is also a valuable metric of that game's popularity, but also ability to stay alive and competitive in the market. Recent tendencies in gaming have demonstrated that resorting to other media outlets (such as movies, series and books) can positively impact the public impression of a game, and drive up the number of people who try it.

- Content and event adherence

  Often more specific to games that provide continuous updates and content over the course of months and potentially years (often referred to as Game As A Service, one of the most popular ways to monetize games), this analytic covers how well received certain sets of content were, propelling decision-making not only from developers but players alike.

2.2.3 Complementing Data with Inquiries

While the aforementioned techniques are useful, it still lacks what statistics usually lacks: an attention to player feel. Statistics are emotionless, and as such may miss some key aspects of the player experience in this incidental negligence to all possible information (El-Nasr, Drachen, and Canossa 2016).

Routine questionnaires are not uncommon to find in online games, either directly or through outside communications like emails. The feedback passed is then used to augment the pre-existing analytical data.

2.3 Research Process

When identifying a problem and proceeding into solving it, one shouldn't base their work on assumptions or unfounded concepts. Working under this foundation will not produce usable or relevant outcomes. Instead, deriving from pre-existing, well grounded, credited knowledge will lead to much more valuable results with solid principles and harder refutation.

In this vein, a literature review is recommended, a process in research wherein the author of a supposition explores and condenses the already accumulated sum of knowledge on the subject at hand, in an effort to properly organize it.

In order to structure and properly conduct the literature review, an adequate method is picked. Between Systematic and Narrative, the former was used, specifically the Systematic Mapping Study methodology. This simplified version of a Systematic Literature Review was chosen due to its adequacy in the topic and surrounding context. It provides a general ample overview of a research area, asserting if evidence is existing and if so, quantifying it, aiding in the identification of gaps.

The first step in this process is the definition of Research Question(s).

The research required to conduct the necessary work to ultimately arrive at the solution should follow a well established methodology, to guarantee a professional organization, in addition to well founded basis and assumptions for the developments. As such, the following Model for how to properly conduct a Research Process was considered (Figure 2.3).

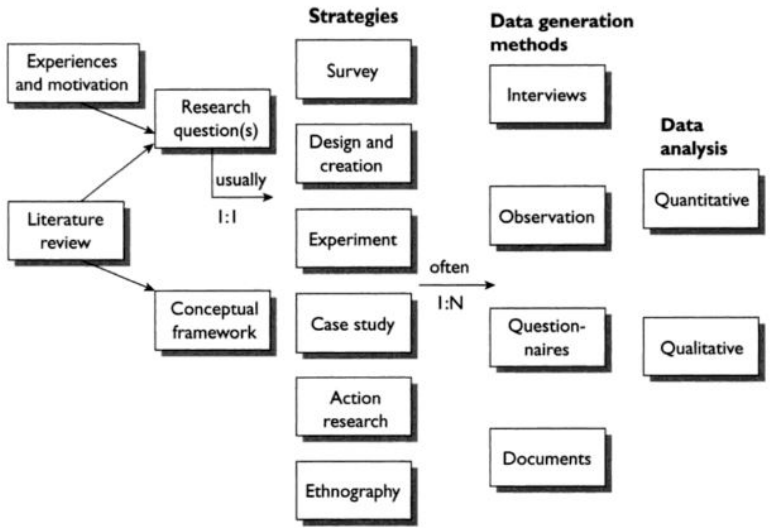


Figure 2.3: Overview of the model for the Research Process and its components (Oates, Griffiths, and McLean 2022).

The proposed model introduces a sequence of activities to be carried out when facing a research challenge, to culminate in an answer. However, the main intent in this method is to properly justify the decisions and findings throughout the process. It defends that it is not enough to arrive at the right or convenient solutions if the audience of the outcome does not accept the discoveries or arguments being made, and therefor do not deem the work worthy of being added to the sum total of knowledge in Information Systems (Oates, Griffiths, and McLean 2022).

As such, firstly the Experiences and Motivations behind the research are pondered. In the past, I have personally played games where the described problem was evident, even if it did not feel like it greatly impacted my experience until I tried to complement it with additional data. Today, I always try to search for more knowledge whenever I find a game I enjoy. Besides, I have always been passionate about games, their impact on people, and trying to

understand the decisions and goals behind them, and as a result have had some ventures with game APIs and other extra functionalities.

This previous contact, not only with the area but also the specific topic, evidences my legitimate interest in this thesis and its surroundings. However, arriving at Research Questions to continue the process was more challenging than expected, which prompted a search on how to more effectively ask these questions and warrant their validity, and this is where the PICOC framework comes into play (Kebede, Moscati, and Johansson 2020).

PICOC stands for Population, Intervention, Comparison, Outcome and Context (*What is a PICOC?* 2023), identified as the five crucial elements of a searchable question. In reality, each of these elements also represent a question, observable in table 2.1. On figure 2.4 an example of its usage is shown, where the method is applied to the topic of self-reporting emotional information (Fuentes et al. 2017).

Table 2.1: PICOC's elements and associated questions

PICOC	**Questions**
Population	What? Or Who?
Intervention	How?
Comparison	Compare with what?
Outcome	What do you want to do?
Context	In which context?

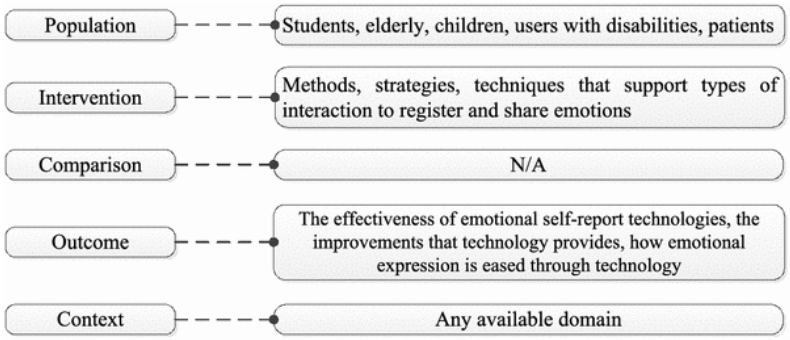


Figure 2.4: Example of Research Questions as structured by the PICOC criteria.

This understanding of the framework was then taken advantage of for intended study, in order to facilitate the definition of Research Questions. In the table (2.2) below, an effort was made toward applying the PICOC method to the topic in study.

Table 2.2: PICOC methodology applied to the topic of using Game Analytics to facilitate players' information processing.

PICOC	Answers
Population	Players of the picked game
Intervention	A new solution that gathers data and automatically interprets it.
Comparison	Officially provided information, existing solutions of the same intent
Outcome	Increase in player performance and engagement; decrease in player effort.
Context	Competitive Multiplayer Online games (namely League of Legends)

Through the usage of the framework, a general research question can be established, guiding the future steps in the research process:

How does one extract and analyse information from a game in order to positively impact a player's performance and experience?

With the research question clearly defined, a systematic literature search was conducted using keywords such as "game analytics", "player performance", "esports analytics", "League of Legends analytics", "Multiplayer online battle arena analytics", "player behavior analysis", and "performance prediction".

The search was performed across multiple academic databases including IEEE Xplore Digital Library, ACM Digital Library, Scopus, Google Scholar, etc. Boolean operators (AND, OR) were employed to combine search terms effectively, such as "game analytics" AND "player performance", "esports" AND "performance prediction", and "League of Legends" AND "analytics".

The systematic search yielded several key studies that directly address different aspects of game analytics and player performance improvement:

A Survey on Game Analytics in Massive Multiplayer Online Games (Fernandes, Castanho, and Jacobi 2018) provided a comprehensive survey of game analytics techniques in massive multiplayer online games, establishing the foundation for understanding how game data can be systematically analyzed.

Comprehensive review and classification of game analytics (Su, Backlund, and Engström 2021) classified game analytics approaches and confirmed that player-focused analytics can guide game design and identify improvement opportunities, directly supporting the premise that game data analysis can positively impact player experience.

Smart kills and worthless deaths: eSports analytics for League of Legends (Maymin 2021) demonstrated that contextualized game metrics provide significantly better insights than raw statistics for understanding player performance, showing that proper data analysis can reveal actionable patterns in player behavior.

Player Behavior and Optimal Team Composition for Online Multiplayer Games (Ong, Deolalikar, and Peng 2015) proved that game data can be used to identify behavioral patterns and predict outcomes, validating that systematic analysis of game information yields practical insights for performance improvement.

League of Legends: Real-Time Result Prediction (Junior and Campelo 2023) achieved high accuracy in real-time game analysis, confirming that game information can be processed and analyzed effectively to provide meaningful insights during gameplay.

The findings from these studies indicate promising directions for using systematic game data analysis to understand and potentially improve player performance and experience, forming a foundation for further investigation in this area.

Chapter 3

Conception & Design

This chapter aims to thoroughly analyze the problem at hand, conceptualizing the approach, identifying the requisites for the solution and determining both what must be done to achieve such requisites and how they should be achieved. It also explores the architectural options available and how the solution can be evaluated by users.

To accomplish these objectives, it begins with a Solution Conception section that aims to provide an early view of the application and the data it will present. It then addresses Design considerations, starting with Requirements Engineering to collect requirements from the stakeholders and their expectations for the final solution structured into Use Cases. These requirements are then analyzed and categorized using the FURPS+ Model into tangible Functional and Non-Functional Requirements. Following this, an extensive breakdown of the design choices to realize the requirements is done in the System Architecture subchapter, sectioned into Backend Design (where the backend architecture, patterns, layers and components are outlined), Riot API Integration (studying the available Riot Games API endpoints, analysing the practical information present in each response and defining the necessary interactions to retrieve data), Game Analytics Design (investigating how the available information from the API can be dissected into statistically relevant data, how to confront it with historical tendencies, and ultimately draw conclusions from comparisons between them), Database Design (where the data persistence layer and relationships between entities are established), Frontend Design (with the intent of delineating a concise and appealing interface for data presentation and improvement recommendations, along with the appropriate flow for backend data communication), Deployment Design (describing the process of containerizing and deploying the application, making it globally available (IMPROVE THIS PHRASE PLEASE), and Survey Design (where the methodology to answer the question posed by this thesis is planned and mapped out, to reach accurate and appropriate conclusions by the end of the study).

This chapter aims to offer an early look into the capabilities of the final solution, aiding in visualizing it, enabling early feedback and highlighting main focuses in design.

3.1 Solution Conception

To provide a clearer picture of the intended result, a projection of its intended form and what it can offer to its users was made. In this projection, a user inputs their in-game name and tag as well as that of any other player they wish to be present, and the system automatically generates a board with the players' ranked information, comparisons, and a set of personal statistics.

Figures 3.1, 3.2, 3.3 and 3.4 illustrate this concept, where a dashboard creation request is made, and data of all players is retrieved through calls to the Riot Games' API using its champion and match endpoints, without resorting to any third-party applications. The data in this example is real and reflects the information for the players at the time of the request.

Figure 3.1: Generic data ranking among several players.

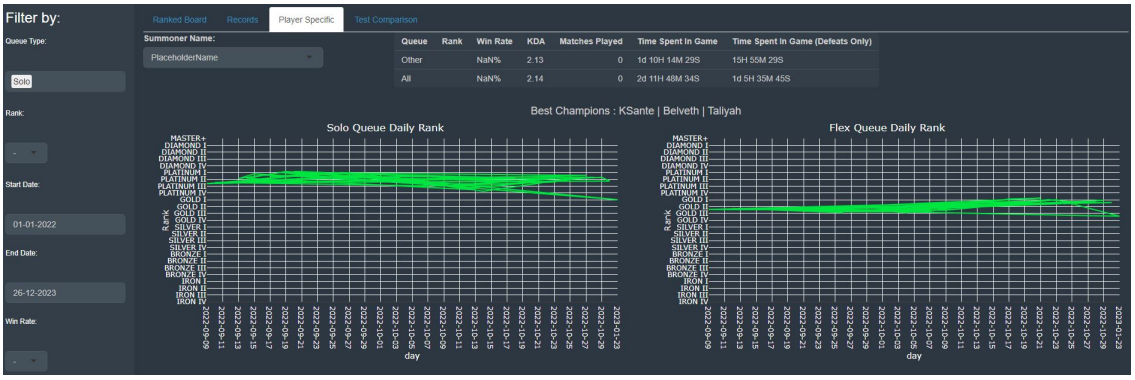


Figure 3.2: More specific data, regarding only one player, documenting ranked evolution throughout the year.

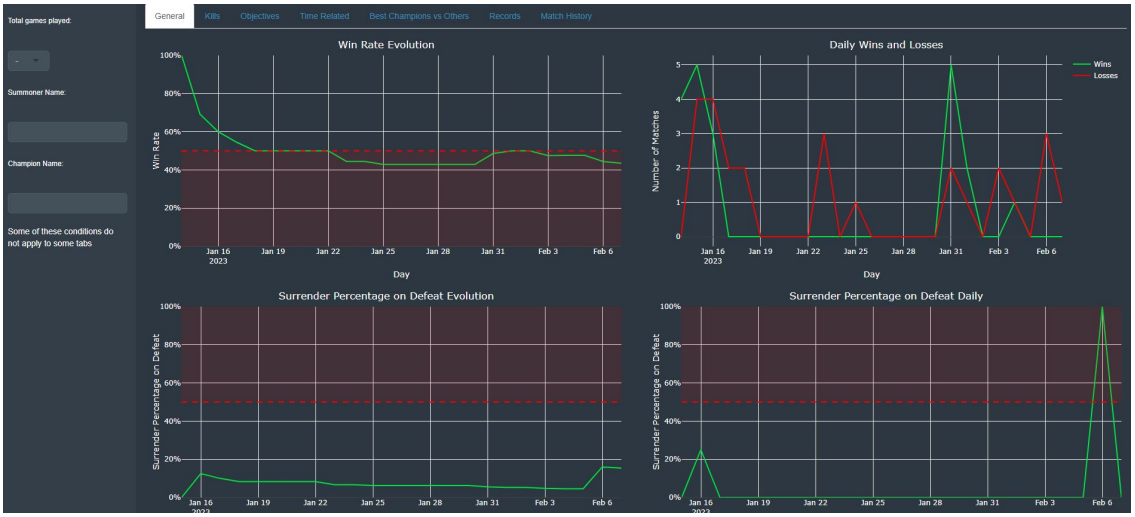


Figure 3.3: Player specific data documenting miscellaneous analytics.

Figure 3.4: Player-specific data with analytics on champion-specific data and performance comparisons between them.

Furthermore, existing solutions were researched and analysed to identify more potentially useful information, trends among data collection applications, and opportunities to innovate in the existing ecosystem. The League Of Graphs website (LeagueOf 2023) contains some valuable assets in this regard, as shown on figures 3.5 and 3.6. It offers relevant insights on player trends and specifics, and draws conclusions from playstyles and tendencies in an easy-to-understand manner.

Figure 3.5: Broad overview of the statistics available.

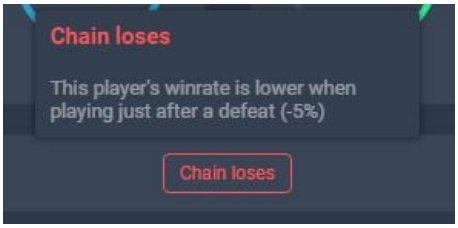


Figure 3.6: Example of the website's player characterizations.

Although both figures show different concepts of what the analytics can be and how to impart information to a user, both offer valuable context to the player, with actionable information to critically assess patterns, improve decision making, and enhance gameplay. Applications like League of Graphs provide a foundation for addressing this thesis' core question, and exemplify what the intended solution seeks to achieve.

3.2 Requirements Engineering

Requirements Engineering is a fundamental phase in the software development life cycle tasked with identifying, documenting, and managing the needs and constraints that a software system must satisfy. This critical process bridges the gap between stakeholder needs and technical implementation, ensuring developers understand the objectives, and the final solution addresses relevant and specific problems effectively, fulfilling its intended purpose. Requirements Engineering, among several activities, mainly encompasses the identification of requirements from stakeholders, the analysis and specification of both functional and non-functional requirements, validation of requirement completeness, and ongoing management of requirement changes throughout the development process.

In the context of software engineering methodologies, Requirements Engineering typically follows the initial problem identification and precedes the system design and implementation phases. Its importance cannot be overstated, as poorly defined or misunderstood requirements are among the leading causes of software project failures Mafra et al. 2016. By establishing a clear understanding of what the system must accomplish and how it should behave, Requirements Engineering provides the foundation for all subsequent design decisions and serves as the primary reference for validating that the implemented solution meets stakeholder expectations.

In the first step of the Requirements Engineering phase for this thesis, the key stakeholders (comprising the company developing the target game and its players) were asked what aspects they saw necessary in an application of this nature. This led to the identification of use cases, which were then depicted in the Use Case Diagram below.

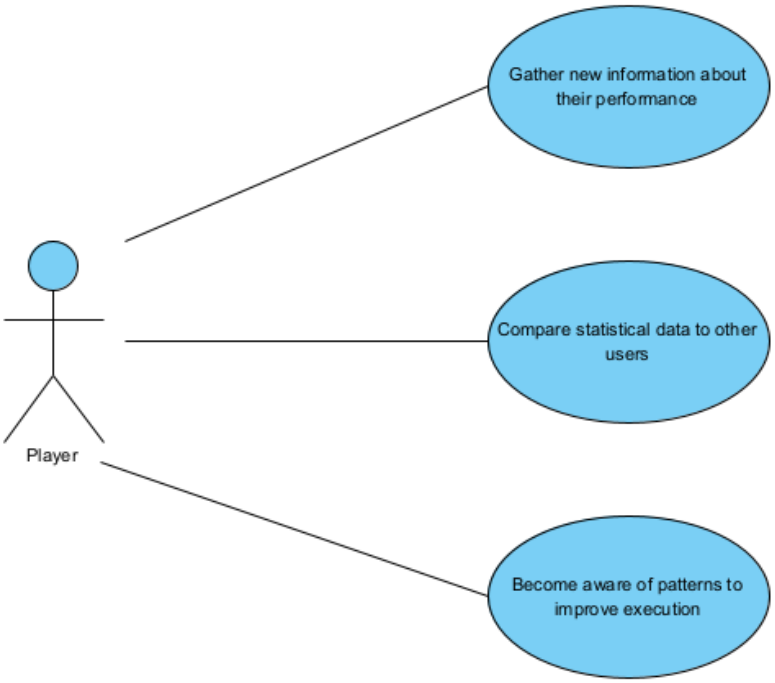


Figure 3.7: Use Case Diagram.

3.2.1 Functional Requirements

Use Cases provide valuable insight into what users want from a system and how they interact with it, capturing goals effectively. Building on this foundation, detailed requirements were identified and categorized through more granular specification, enabling a comprehensive understanding of the system's expected behaviors. To structure this process, the FURPS+ model was employed, a framework for organizing and classifying both functional and non-functional requirements. The first category in the model is Functional Requirements, documented in the table below:

Table 3.1: Functional Requirements.

Requirement	Actor	Description
FR1: Observe recent matches	Player	As a Player, I want to be able to trigger a data aggregation of recent matches for a specific player.
FR2: Gather gameplay data	Player	As a Player, I want to be able to observe a variety of information regarding gameplay patterns, both overall and champion oriented.
FR3: Calculate statistics	Player	As a Player, I want to learn about all non-readily available data. Not just generic, but timing specific, economic, team related, etc.
FR4: Draw conclusions	Player	As a Player, I want conclusions to be drawn from the statistics, in the form of critical feedback, in regards to champion tendencies and equally ranked player comparisons.

3.2.2 Non-Functional Requirements

While Functional Requirements outline what the system must do, Non-Functional Requirements describe how the system should behave in order to carry out the Functional ones. Under the FURPS+ model, they are categorized into Usability, Reliability, Performance, Supportability, and additional concerns surrounding Implementation Constraints and non-functional Security Eckhardt, Vogelsang, and Fernández 2016.

As such, in the aforementioned stakeholder inquiries, the following Non-Functional Requirements were asserted:

Table 3.2: Non-Functional Requirements.

Requirement	Category	Description
NFR1: Easy to understand	Usability	The solution must present data in a very clear and easy to understand way, and conclusions must be actionable.
NFR2: Available	Reliability	The solution must be available 99.9% of the time.
NFR3: Assures Data Integrity	Reliability	The data presented by the solution must be accurate and conclusions trustworthy.
NFR4: Fast	Performance	The solution must conclude analysis as fast as the external dependencies (such as public APIs) allow.
NFR5: Maintainable & Flexible	Supportability	The solution must be easily maintained and modified, as it is dependent on games intrinsically subject to constant change.
NFR6: Respects Limits	Implementation Constraint	The solution must be implemented in a way that respects the rate limits of outside resources such as external APIs, as to not be blocked out.
NFR7: Protects Critical Resources	Security	The solution must never expose any sensitive resources (like API keys) in client-side code, logs, errors, or any other way.
NFR8: Abstains from Private User Data	Security	User data must originate solely from publicly available, authorized game publisher sources.

3.3 System Architecture

After analyzing the requirements that ultimately must be fulfilled, there is now enough context to design the solution. From multiple perspectives, this chapter aims to present the choices and decisions that will guide development, as well as expose potential alternatives and evaluate their flaws in comparison to the chosen approach.

3.3.1 Backend Design

The backend will follow a layered architecture pattern, organized into distinct layers that separate concerns and promote maintainability. This approach will ensure clear separation between presentation logic, business logic, data access, and external service integration, facilitating development and incremental iterations.

The architecture will be comprised of four primary layers, each with well-defined responsibilities:

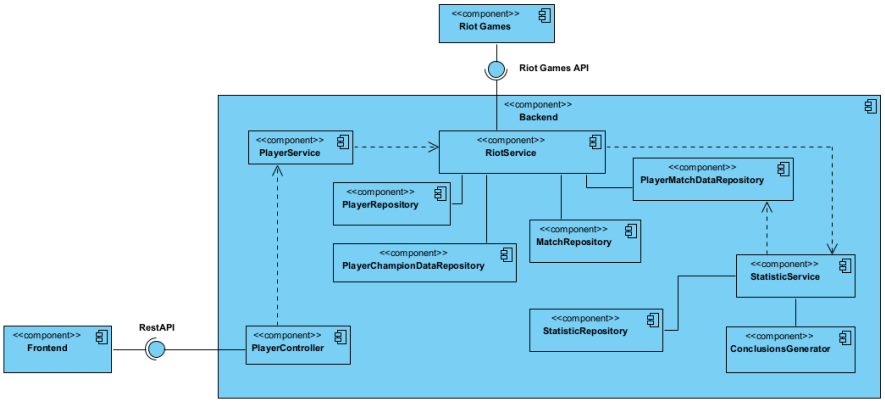


Figure 3.8: Backend Component Architecture

- Controller Layer: will serve as the presentation tier, handling HTTP requests and responses. Controllers validate input parameters, delegate to service layer components, and format responses according to REST API conventions. The `PlayerController` exemplifies this layer's responsibility for request handling without containing business logic.
- Service Layer: will contain the core business logic and organize operations across multiple data sources. Services such as `RiotService` and `StatisticService` will manage data processing workflows, coordinate external API calls, and implement domain-specific algorithms for analytics generation.
- Repository Layer: will provide data persistence abstraction through the Repository pattern, isolating business logic from database implementation details. JPA repositories handle entity persistence, custom queries, and database transactions while maintaining technology independence.
- External Integration Layer: will manage communication with external services, in this context the Riot Games API, through `RiotAPIClient`. This layer is set to handle API calls, rate limiting compliance, error recovery, and data transformation from external formats to internal domain models.

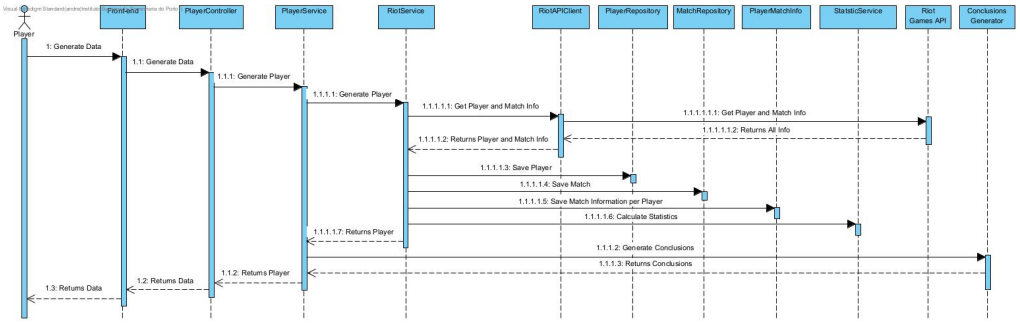


Figure 3.9: Backend Request Processing Sequence

Component interactions will follow the sequence illustrated in figure 3.9. Request processing begins with the Controller layer receiving HTTP requests, which are then delegated to appropriate Service components. Services orchestrate business logic execution, coordinating

between Repository components for data persistence and External Integration components for third-party data acquisition.

The architecture will employ several key design patterns to ensure maintainability and extensibility:

- Repository Pattern: Encapsulates data access logic and provides a uniform interface for data operations, enabling easy testing and database maintenance and updates.
- Data Transfer Object (DTO) Pattern: Separates internal domain models from external API representations, providing stability against external API changes and controlling data exposure.
- Dependency Injection: Promotes loose coupling between components, facilitating testing and component substitution, and ensures the application can easily adapt to revisions.
- Mapper Pattern: Isolates data transformation and aggregation logic in dedicated mapper classes, separating pure computational logic from business orchestration concerns.

This layered architecture will ensure that each component maintains a single responsibility while providing clear interfaces for interaction with other system components. The design supports the system's primary objective of processing game analytics by maintaining clear separation between data acquisition, processing, and presentation concerns.

3.3.2 Riot API Integration

The system will integrate with Riot Games API to acquire real-time player and match data through a dedicated API client component. This integration will serve as the primary data source for analytics generation, requiring a sequence of calls to multiple API endpoints to construct complete player profiles and accurate statistic representation.

The integration architecture will be centered around the `RiotAPIClient` component, which is set to manage all external API communications while abstracting the complexity of rate limiting, error handling, and data transformation from the business logic layer. This design ensures that API-specific concerns remain isolated from the core analytics processing components.

Table 3.3: Necessary Riot Games API endpoints.

Endpoint	**Name**	**Purpose**
ACCOUNT-V1.1	Get Account by PUUID	Translate a combination of Riot Name and Riot Tag into the Player Universally Unique IDentifier.
LEAGUE-V4.1	Get league entries in all queues for a given PUUID	Assert if a Player has recent activity and determine current rank in solo queue and flex queue.
LEAGUE-V4.2	Get all the league entries	Retrieve a list of several players and their ranked general information based on a given division and tier.
MATCH-V5.1	Get a list of Match IDs by PUUID	Retrieve a list of the IDs of the most recent matches for a player.
MATCH-V5.2	Get a match by Match ID	Retrieve end of match information for a given match.
MATCH-V5.3	Get a match timeline by Match ID	Retrieve detailed information of events transpired during a match for a given match.

Table 3.3 presents the essential API endpoints that the system will utilize. The integration follows a structured data acquisition workflow that begins with player identification through ACCOUNT-V1.1, proceeds to activity and rank validation via LEAGUE-V4.1, retrieves match histories using MATCH-V5.1, and finally gathers detailed match and timeline data through MATCH-V5.2 and MATCH-V5.3 endpoints.

Interaction with the API presents specific challenges that the component and application as a whole must address to ensure reliable data acquisition. With a development key, rates are limited to 20 requests per second or 100 requests every two minutes (*Developer Portal* n.d.), and as such there must be a compliance mechanism to prevent quota violations, such as controlled request pacing. The system should also be prepared for error handling in events like invalid responses and routine API unavailability, and regional endpoint selection is mandatory for conformity with Riot Games' requirements.

API Integration will also be responsible for data transformation, from raw Riot API JSON responses to DTOs before returning to the business logic components, assuring loose coupling between external information formats and internal models, making the application more resilient against source data changes.

This integration will support both individual player analysis workflow and bulk data collection operations, taking sole responsibility for external API communications, bolstering the system's capabilities by ensuring reliability and consistency.

3.3.3 Game Analytics Design

The game analytics component represents the core intelligence of the system, transforming raw match data into actionable insights. This component will implement a comprehensive analytical framework that addresses the fundamental research question of whether game analytics facilitate player information processing and performance improvement.

The analytics architecture will be structured around three primary processing stages: data aggregation, statistical analysis, and conclusion generation. Each stage will employ distinct methodologies to ensure that the final insights are both statistically significant and actionable for player improvement.

In preparation for this process, the Riot API League MATCH-V5.1 and MATCH-V5.2 endpoints returns were studied. While most of the information is pertinent, filtering the irrelevant information can only improve the system's performance and guide the metric calculation process and further conclusion assessment.

Fields like `bountyLevel` and `eligibleForProgression` are primarily useful out of the match context, and `nexusLost` and `unrealKills`, while technically pertaining to match data, are not useful for actionable feedback regarding play patterns.

In the Timeline JSON, data is organized in frames, snapshots of the game state at certain intervals (usually of 15 seconds). These can be used in aggregated statistics, calculated throughout a match's existence, ideal for time-sensitive metrics and more granular insight. Here, properties like `position` (map coordinates) and `level` were forgone in favor of more practical data.

3.3.3.1 Data Aggregation Pipeline

The data aggregation stage will process raw match data retrieved from external APIs to generate comprehensive player performance metrics. The aggregation pipeline will compute statistical measures on individual match records stored in PlayerMatchInfo entities.

The system will calculate aggregated statistics spanning both temporal performance indicators and overall game performance metrics. Temporal metrics will focus on early-game performance through fifteen-minute benchmarks, including average kills, deaths, assists, and gold accumulation at the critical fifteen-minute mark, an ending point to micro-oriented beginning of the match. These early-game indicators will provide insights into laning phase effectiveness and early-game decision-making patterns.

Overall performance metrics will encompass full-game statistics including kill participation rates, death contribution percentages, vision score averages, gold per minute efficiency, and creep score per minute measurements. These metrics will be computed using mathematical aggregations that account for variable match durations and ensure statistical consistency across different game lengths.

This aggregation process will be implemented in two scopes, computing both player-wide statistics across all matches and champion-specific statistics for individual champion performance analysis. This will enable contextual analysis that accounts for champion-specific performance expectations and role-based gameplay patterns.

3.3.3.2 Statistical Analysis Framework

The statistical analysis framework will implement a threshold-based comparison system that contextualizes player performance against established benchmarks. The framework will utilize a threshold structure that accommodates multiple levels of statistical comparison: global player averages, rank-specific benchmarks, champion-specific thresholds, and combined champion-rank statistical boundaries.

Statistical significance will be determined through standard deviation analysis, employing predefined statistical boundaries for each performance metric. The system will utilize a `StatBoundaries` enumeration that stores standard deviation values for critical performance indicators, including kill participation, death contribution, vision score, gold per minute, creep score per minute, and early-game performance metrics.

The threshold comparison system will implement a decision framework separated in tiers that categorizes player performance into distinct statistical groups. Performance evaluation will utilize mean-centered analysis with standard deviation boundaries, enabling the classification of player statistics as significantly below average, slightly below average, slightly above average, or significantly above average relative to the appropriate comparison population.

Champion-specific analysis, on the other hand, will employ percentage-based threshold boundaries, utilizing fifteen percent deviation margins to account for champion-specific performance variations. Champion data was found to be more prone to variation, and standard deviation does not accurately complement the intended decision-making framework. This approach will accommodate the inherent performance differences between different champion archetypes while maintaining statistical rigor in performance evaluation.

3.3.3.3 Conclusion Generation System

After threshold comparison, the conclusion generation system will synthesize the analysis results into human-readable, actionable feedback designed to facilitate player information processing and performance improvement. The system will implement the conclusion framework in three categories: champion-based insights, rank-based comparisons, and global player performance analysis.

Champion-based conclusions will compare individual champion performance against champion-specific statistical thresholds, providing targeted feedback on champion mastery and role-specific effectiveness. These conclusions will prioritize champions with sufficient statistical significance, requiring a minimum number of matches to ensure reliable and useful analysis.

Rank-based conclusions will contextualize player performance within their competitive tier, comparing aggregate statistics against rank-specific benchmarks to identify performance gaps or strengths relative to similarly-skilled players. This analysis will provide insights into whether player performance aligns with their current competitive ranking, and improval within this bracket is easily contextualized.

Global player conclusions will evaluate overall performance against universal player statistics, identifying fundamental gameplay strengths and improvement areas that transcend rank or champion-specific considerations. These conclusions will focus on core gameplay mechanics that apply across all competitive contexts.

The conclusion generation process will implement contextual message selection based on statistical band classification, ensuring that feedback tone and content appropriately reflect

the magnitude of performance deviations. Positive reinforcement will be emphasized for above-average performance, while constructive improvement suggestions will be provided for below-average metrics.

3.3.3.4 Analytics Validation

The described analytical methodology is grounded in established statistical principles, and will be validated through data collection from diverse player populations. Thresholds, the key factor in conclusions, will be derived from comprehensive baseline data gathering from multiple sources across multiple ranked tiers, promoting accuracy and reasonable distribution. The minimum match requirement will be a solid quality assurance measure for the conclusions being shared.

This design accommodates future enhancements such as new statistical data from the external API, improved calculation techniques for boundaries and thresholds, machine learning algorithms, etc., and enables consistent improvement through player feedback.

By the end of this process, the complete data retrieval and conclusion generation flow is practicable. The entire procedure is observable in figure 3.10.

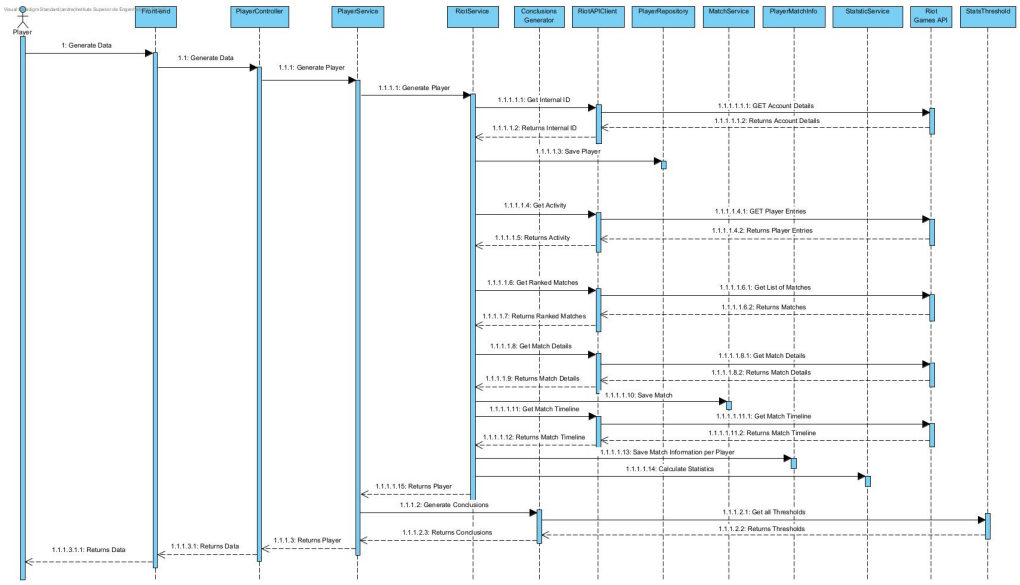


Figure 3.10: Sequence Diagram of the Conclusions Generation Process

3.3.4 Database Design

The database design will follow a normalized relational model that efficiently stores player information, match data, participant statistics, and analytical results while maintaining data integrity and supporting analytical queries. The schema will be structured to accommodate both individual player analysis and aggregate statistical information required for baseline comparisons and conclusion generation.

The architecture will be centered around the `Player` entity, containing all necessary player details and relationships to all of the statistical data. Firstly, it will maintain a one-to-one relationship to the `Statistic` entity and a many-to-one relationship to `Rank`. `Statistic` stores aggregated performance metrics across all matches analyzed, specific to a single

player, while `Rank` represents an aggregation of players of similar skill, and will allow for contextualized comparisons.

For many-to-many relationships, intermediate entities are employed, and a composite key approach will be implemented to connect them with their constituents. The `Player_Match` entity will use a composite key combining player and match identifiers to store per-match performance data, while `Player_Champion_Statistic` will implement a composite key of player and champion to maintain per-champion performance aggregations for each player.

Match data will be structured through the `Match` entity, which will serve as a container for match-specific information and relate to multiple `Player_Match` records representing all participants in that match. This design will support both individual player analysis and team-based statistical calculations.

To enable conclusion generation, the schema will support the threshold system through the `Statistic_Threshold` entity, which will store statistical benchmarks for different combinations of champions, ranks, and global averages. This entity will support nullable champion and rank fields, enabling the storage of global thresholds, champion-specific thresholds, rank-specific thresholds, and champion-rank combination thresholds within a single table structure.

The `Champion` and `Rank` entities can be loaded on application start and will provide storage for game-specific reference information and support foreign key relationships throughout the schema.

Entity relationships will be carefully designed to maintain referential integrity while supporting efficient query patterns. The schema will employ appropriate indexing strategies on frequently queried fields such as player identifiers, match identifiers, and timestamp fields to ensure optimal performance during analytical processing. Figure 3.11 illustrates these relationships.

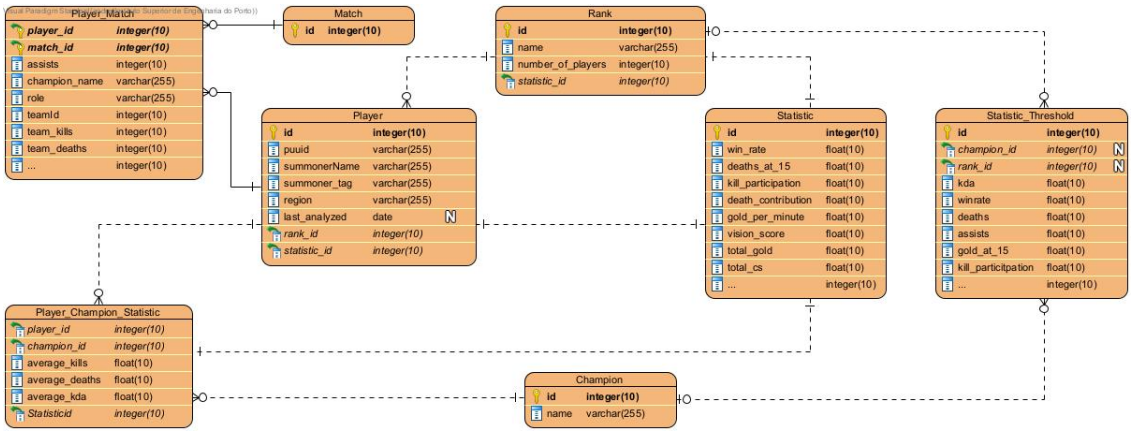


Figure 3.11: Entity Relationship Diagram for the projected solution.

3.3.5 Frontend Design

The frontend will implement a Single Page Application (SPA) architecture using component-based design to provide users with a simple and easy to use experience with concise information. This architectural approach will ensure seamless navigation between different views

while maintaining application state and providing immediate feedback to user actions, all in an appealing and straightforward presentation.

The application will follow a hierarchical component structure organized around core user workflows. The root application component will manage global state and routing, while specialized components will handle distinct functional areas including player search and statistics presentation.

The specialized component section will be comprised of two primary functional areas: the Player Search Component, which will provide the primary user interface for initiating analytics requests through Riot ID input forms in the search functionality and visual feedback of the process status; and the Player Statistics Component, which will present comprehensive player analytics including aggregate statistics, per-champion performance metrics, and generated conclusions in an organized, readable format.

The frontend architecture will employ a service-based approach for data management and API communication. The `PlayerService` will handle all backend API interactions, abstracting HTTP communication details from presentation components. This separation will ensure that user interface components remain focused on presentation logic while data acquisition concerns are centralized in dedicated service classes.

Navigation will be implemented through a routing system, providing bookmarkable URLs and browser history support. The routing structure will support both direct navigation to specific player profiles and guided redirecting through the search interface.

Most importantly, the user interface design will prioritize clarity and actionability, presenting analytics data in digestible formats. The interface will emphasize constructive feedback presentation, ensuring that the generated conclusions are clearly distinguished from raw statistical data. Statistical information will be organized appropriately, with summary-level insights prominently displayed while allowing the discernment of detailed metrics. The design is set to be responsive, ensuring an optimal user experience across different device types and screen sizes, with particular attention to mobile accessibility.

This design aims to provide a pleasant, responsive, straightforward experience to the user, preventing confusion and incentivizing the participation of this study.

3.3.6 Deployment Design

The deployment architecture will employ a containerized approach to ensure consistent environments throughout development and production while facilitating scalability and maintenance. The system will be deployed as a multi-container application, providing clear separation of concerns between application tiers.

The deployment will comprise three primary containers: a database container for data persistence, a backend container for API services and business logic, and a frontend container for static content delivery and client-side application serving.

In addition, container orchestration will enable coordinated startup, networking, and environmental variable management for flexibility and consistency. The frontend container will serve as the primary entry point into the app's environment, handling client requests and routing API calls to the backend container, which will in turn interact with the database container in data operations in the conclusion generation procedure.

The containerized environment will be deployed to web server infrastructure, with traffic routed through a content delivery and proxy service for enhanced security, and global distribution. Figure 3.12 illustrates this architecture.

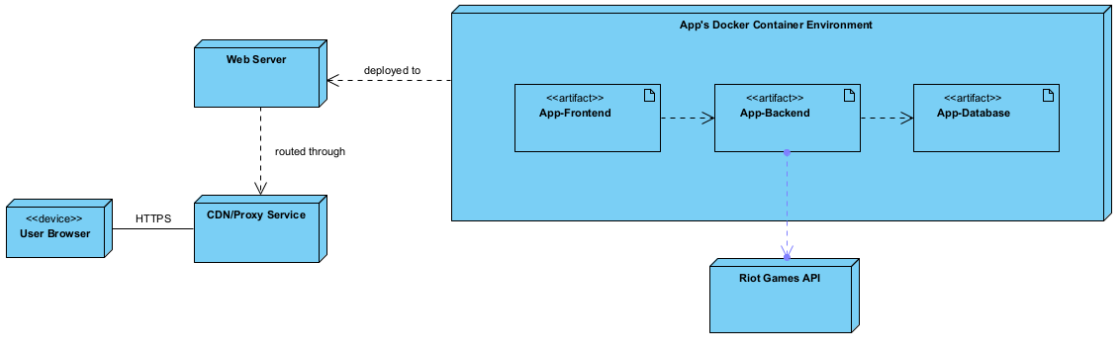


Figure 3.12: Deployment structuring for the suggested application.

Chapter 4

Implementation

This chapter presents the practical development of the game analytics system designed in the previous chapter. The implementation translates the architectural specifications and design decisions into a fully functional web-based application capable of retrieving, analyzing, and presenting League of Legends match data to users.

The development process encompassed several key areas: selecting and implementing the technological stack, establishing communication with Riot Games' API, developing the system architecture with both backend and frontend components, implementing robust data handling and processing mechanisms, and designing an intuitive user interface and experience.

The following sections detail the technology choices made, the challenges encountered during API integration, the realized system architecture, data processing workflows, and the user interface design that enables players to access contextualized game analytics effectively.

4.1 Technological Stack

4.1.1 Back-End

The back-end of the solution was built using Java as the main language. Its object-oriented nature, combined with the array of frameworks and utilities available, made it a suitable choice in an application that anticipated robust data processing needs and standard API development practices. Design constraints like NFR5 (Maintainable & Flexible) or external API dependency were also factored when making this choice.

The Java ecosystem is leveraged through the adoption of the Spring Boot framework for added functionality and speed (native support for RESTful architecture, faster development, dependency injection, among others) and with the usage of Hibernate as the Object-Relational Mapping solution, providing vigorous support in database interactions and smooth entity management.

For dependency management and build automation, Gradle with Kotlin DSL was used. While Maven was considered, it was ultimately forgone, since the objective is a fast, performant, flexible application (as expressed in NFRs 4 and 5, namely Fast and Maintanable & Flexible), and Gradle aligned better with our goals.

4.1.2 Database

Database configuration initially started with H2 as a very basic, lightweight, and easy to set up in-memory option for early development and testing. As the solution's capabilities and features kept growing, and the application evolved to handle more complex data relationships and require persistent data storage, PostgreSQL replaced H2 as a highly concurrent alternative capable of handling bigger workloads, more scalable and with built-in support for replication, ready for production level requirements.

4.1.3 Front-End

To achieve the intended simple, yet engaging, single-page application, the front-end was developed using the Angular framework with TypeScript as the primary language. This allows for a responsive and modern user interface while ensuring the code is highly maintainable and adjustable.

Angular's ecosystem enables the usage of libraries that make the anticipated design challenges much more manageable, such as Angular Router for fluid navigation between components and states, Angular HttpClient for reliable and efficient API communication management with the back-end, and Angular Material for a consistent and appealing user interface.

4.1.4 Version Control & Deployment

GitHub was used for Version Control throughout the project's development, with Jira integration, using clear branch division assigned to incremental tasks (for example, "Merge all backend endpoints into an all-purpose one").

On the Deployment side, containerization is done through Docker, and publishing is achieved with a local server monitored and managed through Portainer (as evidenced below in figure 4.1), also allowing deployment and update directly from Git.

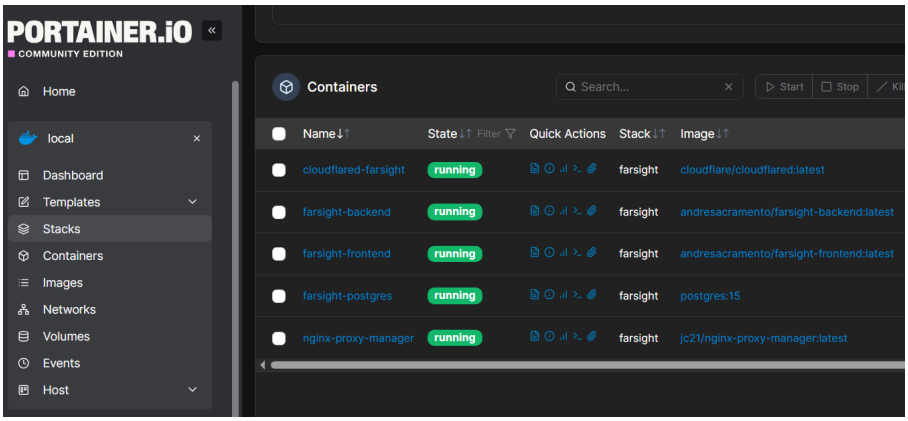


Figure 4.1: The Farsight stack on Portainer.

Additional measures were adopted, like reverse proxy through Nginx (and Nginx Proxy Manager for easier configuration and management), and Cloudflare integration for several security actions like DDOS protection, web application firewall, rate limiting and many more, and access to their Content Delivery Network and Cloudflare Tunnel. The Cloudflare Content Delivery Network allows for quicker website functioning overall (asset transfer, loading,

requests) by accessing the closest server in a global network of geographically distributed locations. It also helps to protect against some common attacks *What is a CDN?* n.d.

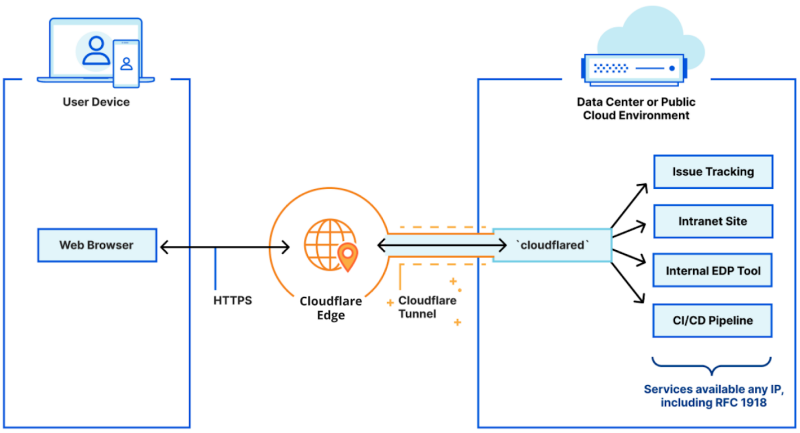


Figure 4.2: How a Cloudflare Tunnel works.

Cloudflare Tunnel, pictured above in figure 4.2, focuses solely on security, allowing the exposition of a container inside a machine instead of the machine itself. In a normal setup, the port that is proxied through Cloudflare is still exposed and vulnerable. Through the Tunnel, ports or public IPs are not exposed to outside traffic, which is forced to go through the Cloudflare Edge, severely increasing the safety of the application and diminishing the chance and impact of attacks *Getting Cloudflare Tunnels to connect to the Cloudflare Network with QUIC* 2021.

4.2 Riot API Communication

The implementation of the projected solution relied heavily on communicating with Riot Games API, as it is the only publicly available direct source of player information for League of Legends. After acquiring a basic development key, the first steps in implementing the solution were establishing infrastructure for all requests to the API.

In the API, a player can be identified by either their public identity (for League of Legends this is a combination of Riot Name and Riot Tag, assigned to a Region) or their private, Riot-specific identity, denominated Player Universally Unique IDentifier, or PUUID for short *Player Universally Unique IDentifiers and a New Security Layer* 2018. PUUIDs are Riot's in-house solution for a consistent, individual identifier spanning their various games and systems, and while it still allows a user to change their public identity, it is immutable to its respective player, ensuring it is globally unique. *PUUIDs and Other IDs* 2019

While the PUUID is personal, it is not hidden, and with the exception of querying precisely the account information to uncover the PUUID, all endpoints for a specific player depend on this private identifier. Moreover, users are supposed to query their own information, and they are not expected to know this seventy-eight alphanumeric randomly generated identification to do so, especially when the public identity is available. As such, the first implemented functionality for the solution will be a query to endpoint ACCOUNT-V1.1, "Get Account by

PUUID". This is also the opportunity to mainstream the process of all requests to the Riot Games API.

At first, the basic structure was created according to the Design: Controller, Service, Repository and API packages, followed by RiotRequestsController, RiotService interface and implementation, and a RiotAPIClient class, all properly stored. The controller is responsible for being the entry point for communications and making the solution's endpoints available, and those were precisely the first steps in the application's development, as pictured below on figure 4.3. Then, the service is behind business logic, acting as a bridge between the controller and calls to the Riot Games API endpoints, starting with very basic code as depicted in figure 4.4.

```
import ...

@RestController  AndreSacra +1 *
@RequestMapping("riot-coms")
public class RiotRequestsController {

    RiotService riotService;  2 usages

    @Autowired  André Sacramento *
    public RiotRequestsController(RiotService riotService) {
        this.riotService = riotService;
    }

    @GetMapping(value = "puuid")  new *
    public String getPuuid(@RequestParam(value = "name") String name,
                           @RequestParam(value = "tag") String tag) throws Exception {
        return this.riotService.getPuuid(name, tag);
    }
}
```

Figure 4.3: The first endpoint created, getPuuid.

```java
@Service  AndreSacra +1 *
public class RiotServiceImpl implements RiotService {

    private final RiotAPIClient riotAPIClient;  2 usages

    public RiotServiceImpl(RiotAPIClient riotAPIClient) {  André Sacramento *
        this.riotAPIClient = riotAPIClient;
    }

    /**
     * @param name The player's username
     * @param tag  The player's tag
     * @return The player's PUUID (Player Universally Unique IDentifier)
     * @throws Exception
     */
    @Override  1 usage  new *
    public String getPuuid(String name, String tag) throws Exception {
        return riotAPIClient.getPuuid(name, tag);
    }
}
```

Figure 4.4: Service Implementation for the getPuuid call.

While the external call could technically have been done directly on the service, the Design phase highlighted the high volume of potential interactions between the back-end portion of the solution and the Riot Games API, and as such a dedicated class is much more suitable, scalable and maintainable. As mentioned, the intent was to systematize the call process from the very beginning of development. To this end, the service invokes the RiotAPIClient class, which has dedicated methods for building the URLs for communication with Riot, and generic methods for calling the URLs and retrieving the intended information.

Calls to the Riot Games API require an API Key: an authorization and identification mechanism that enforces rate limits, allowing Riot to understand the source of requests and securely respond to them. Since general development keys have a twenty-four hour lifespan and all calls need a key, Dependency Injection was implemented into the Client class to dynamically load the key value from the configuration files. The `@Value` annotation is used to inject the key from the `application.properties` file at runtime, easing the daily update process without code adjustments or dependency on hard coded values.

This is then appended to the base URL for PUUID retrieval, along with Riot Name and Tag, as observed in figure 4.5, and the full URL is passed to the extraction-dedicated method. At this stage the method is very basic, verifying the success of the call to the full URL, extracting the value of the intended property (in this case the `'puuid'`) and returning it, throwing an exception in the event of an error, as evidenced in 4.6. This method can be used on every instance of an API call to Riot with the intent of extracting a single specific value from a property.

```java
@Component 3 usages  André Sacramento +1 *
public class RiotAPIClient {

    @Value("${API_KEY}")
    private String apiKey;
    private static final String APPEND_KEY = "api_key="; 1 usage

    private final RestTemplate restTemplate; 2 usages

    public RiotAPIClient() {  AndreSacra
        this.restTemplate = new RestTemplate();
    }

    /**
     * Get Account by PUUID endpoint
     */
    public String getPuuid(String name, String tag) throws Exception {  1 usage  new *
        String fullUrl = "https://europe.api.riotgames.com/riot/account/v1/accounts/by-riot-id/" + name + "/" + tag + "?" + APPEND_KEY + apiKey;
        return extractFromResponse(fullUrl);
    }
```

Figure 4.5: Preparing the URL, injecting the API Key, for call and data extraction.

```java
private String extractFromResponse(String url, String toExtract) throws Exception{  2 usages  André Sacramento +1 *
    try{
        ResponseEntity<String> responseString = restTemplate.getForEntity(url, String.class);

        if(responseString.getStatusCode() == HttpStatus.OK){
            String responseBody = responseString.getBody();
            JSONObject jsonObject = new JSONObject(responseBody);
            return jsonObject.getString(toExtract);
        } else {
            throw new Exception("API request failed with status: " + responseString.getStatusCode());
        }
    } catch (JSONException e) {
        e.printStackTrace();
        throw new Exception("Error parsing JSON.");
}
```

Figure 4.6: Preparing the URL, injecting the API Key, for call and data extraction.

With the PUUID all other information retrieval endpoints become available. The next step was to implement the LEAGUE-V4.1 endpoint, "Get league entries in all queues for a given PUUID", to determine a player's rank in the existing competitive queues, while asserting if they are active (have recently engaged with the game): if a player is inactive, the process can halt, saving resources and improving the performance of the solution.

The same process as before was employed: create a new `isActive` endpoint on the Controller, calling the Service, which calls the Client. As mentioned before, the PUUID retrieval would be the only server-agnostic operation, whereas all others require some form of routing. The table 4.1 below shows the different League servers, their API route, and their respective region.

Table 4.1: Region Routing for each endpoint

Server	Route	Region
North America	NA1	AMERICA
Brazil	BR1	AMERICA
Latin America North	LA1	AMERICA
Latin America South	LA2	AMERICA
Europe West	EUW1	EUROPE
Europe Nordic	EUN1	EUROPE
Middle East	ME1	EUROPE
Russia	RU	EUROPE
Turkiye	TR1	EUROPE
Korea	KR	ASIA
Japan	JP1	ASIA
Oceania	OC1	SEA
Southeast Asia	SG2	SEA
Taiwan	TW2	SEA
Vietnam	VN2	SEA

This separation was set up in a `RegionMappingService` class initialized with two maps: one linking the route key to a region value, and the other linking the route key to a server name value. When a player queries their data, inserting their Riot Name, Riot Tag and League Server, this class assigns the adequate route for LEAGUE-V4 requests, and the correspondent region for MATCH-V5 requests.

With region mapping handled and applied in the URL construction phase of `isActive`, the call is made to the LEAGUE-V4.1 endpoint. If the return is an empty JSON, the player is set as inactive and the method returns a `false`; if there's content on the resulting JSON, the player's ranked information is extracted by accessing the object with the `queue` property set to either 420 (solo queue) or 440 (flex queue) and stored, and a `true` is returned.

While the first requests to the API were focused on user information, the third endpoint, `getRankedMatches`, aimed to fetch the list of recent matches for a player, iterating over it

and filtering for ranked matches. The very same structure was applied up till the `RiotAPIClient` class, where server routing was done according to region instead of route (all MATCH-V5 calls are routed through the region). The returned JSON is a simple list of match IDs that the application then iterates over, checking the match type through the `getMatch` service described below, and returning the list of the ranked ones.

The `getMatch` endpoint is a straightforward call to the MATCH-V5.2 counterpart on Riot's API, immediately mapping the JSON response according to a `MatchWrapperDTO` if the match type is ranked, and lastly `getMatchTimeline` is created to call MATCH-V5.3 specifically for ranked matches only, retrieving a file with information on game state events at every fifteen seconds (denominated internally as "frame"), mapping it directly to a `TimelineWrapperDTO`.

With all Riot Games API interactions built and ready to use, all five endpoints were merged into a single all-purpose resource, `generatePlayer`, executing each of the previously described steps sequentially and automatically, transforming an input of Riot Name, Riot Tag and League Server into a player instance, several match records, and a collection of statistical data ready for processing and analysis.

4.3 System Architecture

4.3.1 Backend Structure

The layered architecture established in the Design chapter was implemented by organizing the code in clearly defined packages, in a structure that enforces separation of responsibilities. The classes in the `controller` package contain the REST endpoints of the solution, handling incoming requests and delegating to the appropriate `service` classes, which in turn are responsible for the business logic and coordination with the necessary `repository` tasked with interacting with the classes in the `entity` package, the domain models representing core business objects. `DTOs` found in the `dto` package represent the responses from the external API communications made in `api` and bridged them with the internal standards, using the suitable `mapper` for data transformation logic. Additional concerns are addressed through the remaining packages: `config` for application configuration and data setup, `util` for recurring utility functions and `exception` for error handling.

4.3.2 Stats Request and Conclusions

According to Design 3 the player is meant to request their analysis with a single button press, and as such the endpoints laid out on 4.2 were merged into a single all-purpose endpoint which sequentially triggers all the necessary communications, transforming an input of Riot Name, Riot Tag and League Server into a player instance, several match records, and a collection of statistical data ready for processing and analysis. The process is pictured on the Sequence Diagram below, on figure 4.7.

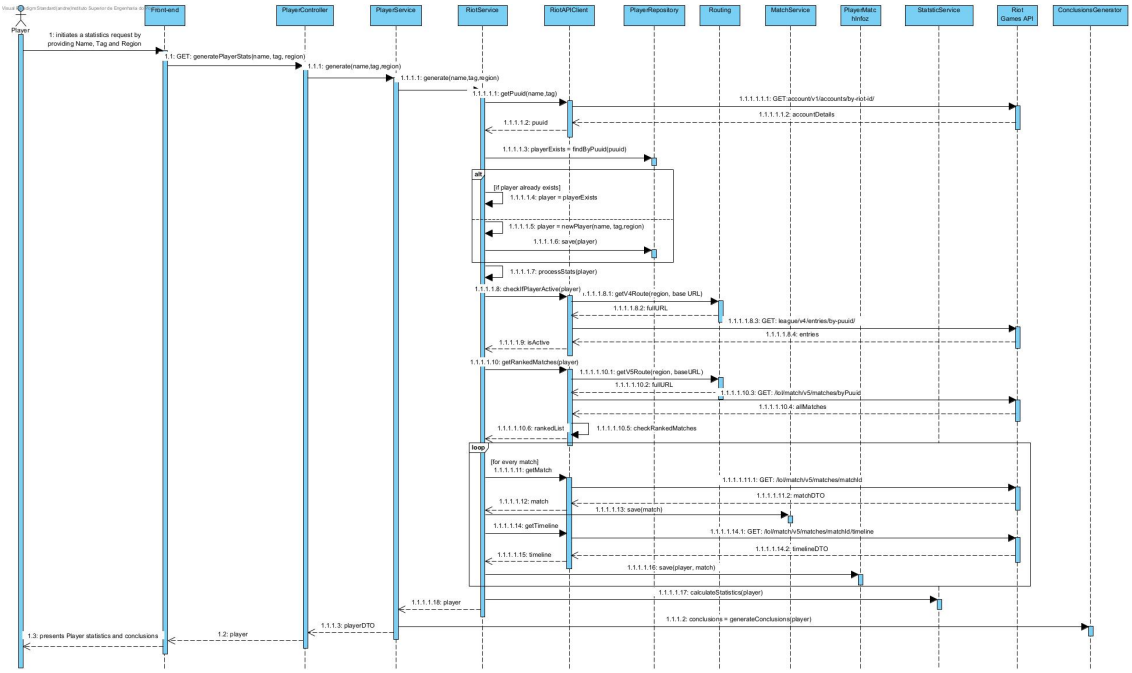


Figure 4.7: The `generatePlayerStats` flow Sequence Diagram.

4.3.3 Frontend

The front-end service was outlined as very simple and smooth experience, and as such implementation followed a Single Page Application pattern with component-based principles. There are two main components in this system: `player-search` for user input and initiation of the analysis process, sending the necessary parameters to the back-end service, and `player-stats`, where the app is routed to when an analysis is successful, displaying statistics and calculated metrics. Each of them is adaptive and responsive, with reactive forms for data handling (like category expansion on the player feedback tab) and user input validation (like the real-time sanitization on the Riot Tag), and transition between the modules is seamless thanks to the `AngularRouter` routing mechanism, which allows navigation between views without full page reloads.

To communicate with the back-end it follows a service-oriented pattern, where dedicated services like `PlayerService` abstract HTTP operations for maintainability. `TypeScript` interfaces are employed,like `PlayerDTO`, to promote type safety and maintain consistency between back-end and front-end data models. The app provides status transparency in several ways, with service-level error transformation to display user-friendly messages (in place of the original raw HTTP errors), and component-level feedback by visually displaying the progress of application state, be it analysis, queue or error.

One of the main concerns of the front-end serve was complementing this architecture with a suitable user experience, aiming for an appealing, light, and easy to comprehend interface. To this end, `AngularMaterial` was used to ensure visual consistency across all `Angular` elements, with MatCardModule providing structured layouts, MatFormFieldModule handling form validation feedback, and MatProgressBarModule offering visual progress indication during operations.

4.3.4 Frontend-Backend Integration

The primary integration point between the front-end and back-end services utilizes a reverse proxy implemented through `Nginx`, a pattern that abstracts the back-end access point from the front-end by intercepting requests and redirecting them to the appropriate destination, while enabling caching and load-balancing capabilities. Additionally, `Nginx` handles Cross-Origin Resource Sharing (CORS) compliance, a browser security feature that prevents unauthorized access from unknown domains by enforcing same-origin policy. Since the front-end and back-end were implemented as separate services, CORS would block cross-origin requests; the reverse proxy approach centralizes this by adding the proper CORS headers to all responses, eliminating the need for application-level configuration.

4.3.5 Jobs, Queue System and Polling

The system implements an asynchronous job queue mechanism to handle long-running player analysis operations, guaranteeing a responsive user experience while limiting the number of concurrent analyses to prevent breaching API rate limits. When a player analysis is initiated, the backend creates an `AnalysisJob` entity with a unique identifier and PENDING status, immediately returning the job ID to the front-end without blocking the request thread. The analysis process runs asynchronously, updating job status from PENDING (awaiting its turn to initiate analysis), to PROCESSING (data retrieval and analysis in progress), to finally COMPLETED or FAILED (according to the result of the operation), with queue position tracking and estimated wait time calculations for user transparency.

To supplement the queue mechanism, the front-end implements a polling procedure that periodically queries the job status endpoint every 3 seconds, preventing `Cloudflare`-mandated request timeouts and providing real-time feedback to users about analysis progress. This approach decouples the user interface from the back-end processing time, allowing the system to handle multiple concurrent analyses while keeping the status information up-to-date. The polling continues until job completion or failure, at which point the front-end either navigates to results or displays case-specific error messages, demonstrating effective separation of concerns between user experience and back-end processing capabilities.

4.3.6 Deployment

All components were containerized using `Docker` to ensure reproducibility, version control, and consistency across environments. Integration was achieved through `Docker` Compose orchestration, which defines a dedicated network that, in this solution, connects the back-end, front-end, and database services, simplifying dependency management and deployment, while centralizing configuration and isolating resources. The Docker images are then uploaded to the `Docker` platform, and deployed in a `Portainer` stack, where they are orchestrated, managed, and monitored. Production deployment extends this architecture, with configuration for a `Cloudflare Tunnel` for added security in external accesses and `Nginx Proxy Manager` for SSL termination (offloading traffic encryption) and enhanced reverse proxy capabilities.

4.4 Data Handling and Processing

4.4.1 Data Collection

The data collection procedure begins when the match information is being extracted in the `getMatch` method in `RiotAPIClient`. At this point, the match has been confirmed to be ranked, and the response from the external API is filtered to retain only the relevant values, identified in chapter 3. The Mapper class directly converts them to a `PlayerMatchInfo` object, referent to the individual statistics for a specific player in that specific match, which is then persisted and updated on the `getMatchTimeline` method with metrics calculated along the recorded frames, like crucial early game performance indicators (for instance, number of assists at the 15 minute mark).

With all matches and match timelines processed, the collection of `PlayerMatchInfo` records for a player constitutes all the available information to assess their statistical details, particularly for the picked metrics. This is set in motion by invoking both the `StatisticService` and the `PlayerChampionStatisticService`, which through their calculation methods transform `PlayerMatchInfo` entities into meaningful, player-specific and champion-specific values, a statistical summary of key indicators like gold per minute and win rate. This results in all Statistics for that particular player, laying the groundwork for Conclusions.

To have a comparison basis for initial analyses and build a foundation for the construction of Conclusions, the process described above was executed for two hundred and fifty players of each division of each rank below Master, by accessing the `LEAGUE-V4.2` endpoint, extracting the PUUID in each DTO, and running the Statistics retrieval for every one of them. In other words, the data collection was executed for a group of seven thousand players, to gather a good outset of the average for each Statistic. Although this was appropriate for general metrics, League of Legends is comprised of one hundred and seventy one champions, meaning champion-specific Statistics could end up being skewed, and to overcome this limitation two of the main analytics-gathering websites were queried for their data on numeric tendencies for champion data in ranked games, with more than seven thousand games sampled, resulting in a very thorough baseline and a good assertion of threshold values for champion Conclusions.

To conclude, Statistics are then compared with the reference and threshold values to determine the Conclusions for the player, the analytical feedback to their performance, play patterns and trends. All of the interactions are carefully persisted through the `Player`, `Match`, `PlayerMatchInfo`, `Statistic` and `PlayerChampionStatistic` entities, as evidenced in the entity relationship model.

4.4.2 Limitations, Efficiency, and Errors

To comply with Riot Games API's constraints, forced rate-limiting mechanisms were implemented at the configuration level using Spring's `@Value` annotation, injected through properties files and/or image environmental variables. They operate at two points of the communication with the API:

- `IS_RANKED_DELAY` forces a delay between ranked match verification calls. After several stress tests, it was settled at 1750 milliseconds;
- `GET_MATCH_DELAY` forces a delay between match data retrieval operations, both regular match information and match timeline information. After careful and extensive testing, it is set at 750 milliseconds.

This ensures the behavior of the system obeys the API's limits at all points, regardless of the current workload. However, the API itself, like other of its size, is inherently prone to throwing 502 Bad Gateway errors due to its high traffic nature (inevitability of one request among millions arriving at a briefly dead gateway, returning a failure), rate limiting configurations (too many requests arriving at once can cause an error response even if it was done properly), or just routine instability. To mitigate this, a retry logic was constructed, with progressively incremental intervals between retries: upon receiving a Bad Gateway error, the application waits two seconds and repeats the call; if the same error is returned, it repeats the process but with a four second deferral; and finally, if a 502 is once again thrown, the system waits eight seconds before sending a last attempt at a successful request. If it still fails, the unavailability of the API is more than temporary and no further calls are made. This promotes a sustainable and consistent interaction between the app and the external API.

Apart from 502 Bad Gateway, there are other errors that are commonly thrown in regular exchanges with the API. In `RiotAPIClient` and `PlayerController`, comprehensive exception handling and message transformation are implemented to address standard scenarios and clearly convey to the user the cause of the error:

- Forbidden Access: API calls may be denied due to a malfunction on the application (like sending too many requests, breaching the rate limits), failure on Riot Games' side, or change of access policy in the resources being accessed. In the event that this happens, a 403 Forbidden is thrown, with custom message `"API access denied. The service may be temporarily unavailable."`;
- Player Not Found: Users may input non-existent combinations of name and tag, or the wrong region for said combination. The API will return a 404 Not Found, which is interpreted and transformed to a more informative version, `"Player not found. Please check the name and tag."`;
- Inactive Player: when a player is identified as inactive, either by the return of LEAGUE-V4.1 endpoint being empty or having no ranked games in their recent activity, the input was technically valid at first, but the absence of further data means that the data retrieval process cannot be executed. As such, while valid at transport level, the request is unprocessable at the domain level, in the context of the service's functional requirements. The application throws a 422 Unprocessable Content, with message `"This player appears to be inactive or has no recent ranked games."`;
- Generic Error: if any other problem is found during the process, a generic `"Analysis failed. Please try again later."` message is returned.

For consistency and clarity, error messages are persisted on the Analysis Job records, and the `@Transactional` annotation is used on multiple Services for automatic rollback capabilities as a prevention mechanism in case of processing failures, to avoid partial data states.

To further optimize the performance of the application, reduce unnecessary API load, and enhance user experience, several steps were taken. Among them, a timestamp-based caching mechanism was put in place, saving when a player was analyzed for the last time in the `lastAnalyzed` field: if this happened in the last 12 hours, the analysis immediately skips the queue process, redirecting the user immediately to the results screen. Furthermore, record duplication checks are made consistently throughout the entire procedure, preventing

overlapping analysis requests and database entries, and API overload. All calls also make use of JPA Repository methods with custom queries for appropriate database interaction.

Finally, the amount of data persisted had the potential to scale very rapidly, and encumbering users with the responsibility to request their own data deletion is inconsistent with the simplistic and easy to use nature of the app. As such, an automatic process was put in place, where players were organized by date of creation and the aforementioned last analysis timestamp, and terminated from the database if their most recent operation exceeded a one-month lifespan.

4.5 User Interface and Experience

The user interface implementation prioritizes simplicity and clarity to ensure players can quickly access their analytics without technical barriers. The application follows a single-page application (SPA) architecture with two primary screens: a streamlined search interface and a comprehensive results dashboard. The overall visual design is kept consistent and appealing through the use of `Material Design` and adherence to a cohesive and vibrant color scheme.

4.5.1 Search Screen

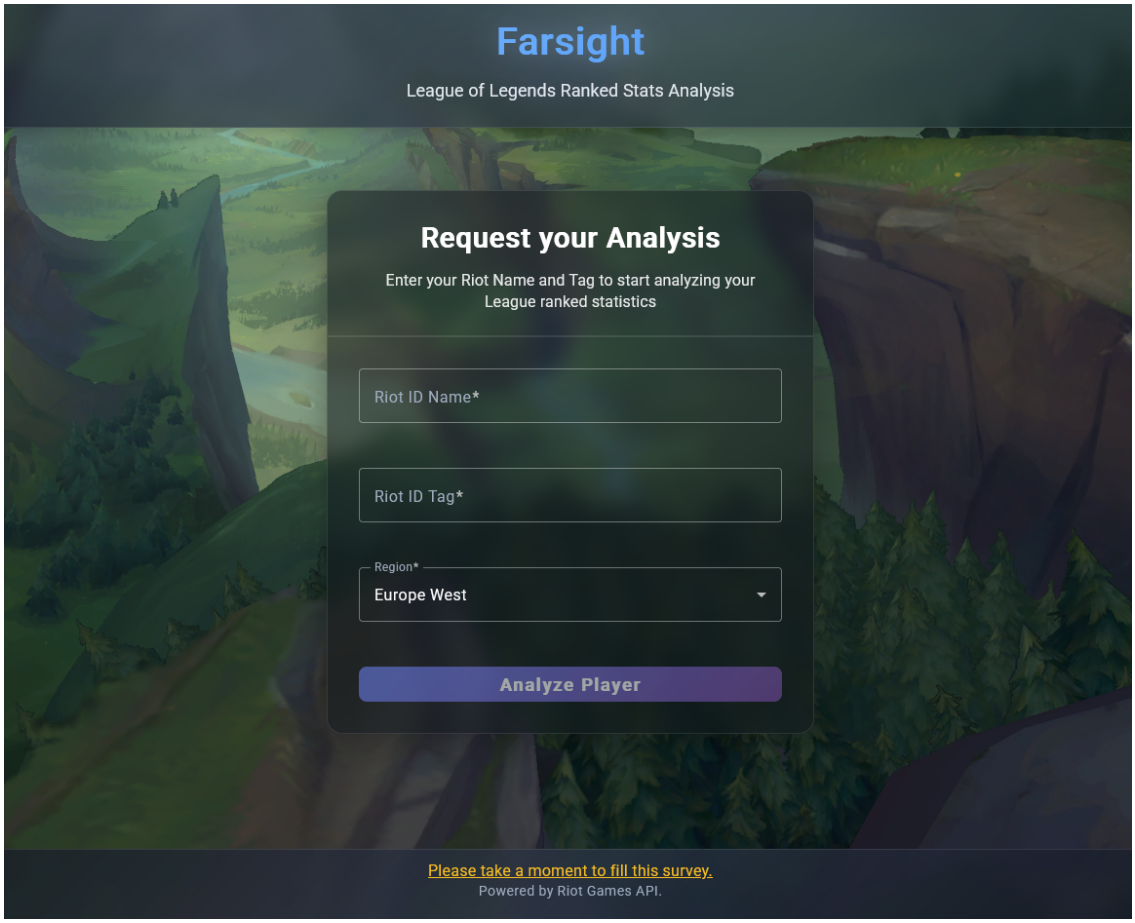


Figure 4.8: The front page of the front-end service, entitled Farsight.

The search screen (pictured in Figure 4.8), the front page of the application, implements a minimalist design with a single form containing a field for each of the three essential inputs: player name, tag, and region selection. The simplicity in this screen is intentional, reducing the imposed mental effort of interpreting new information and eliminating potential user confusion, as human-centered design principles emphasize minimizing users' cognitive load by supporting performance and avoiding unnecessary distractions (Oviatt 2006): for the player, the only objective at this stage is requesting an analysis.

User experience considerations include real-time input validation with immediate visual feedback, automatic tag sanitization to prevent common input errors (such as inserting the hash symbol that separates Name and Tag in the full name), region selector drop-down of the recognizable version of region names (like "Europe West" instead of "EUW1", to accommodate users unfamiliar with Riot's technical naming conventions), and a prominent loading state during analysis initiation. The interface provides clear, intelligible error messaging for invalid inputs or inactive players, ensuring users understand what went wrong and how to proceed.

4.5.2 Results Screen

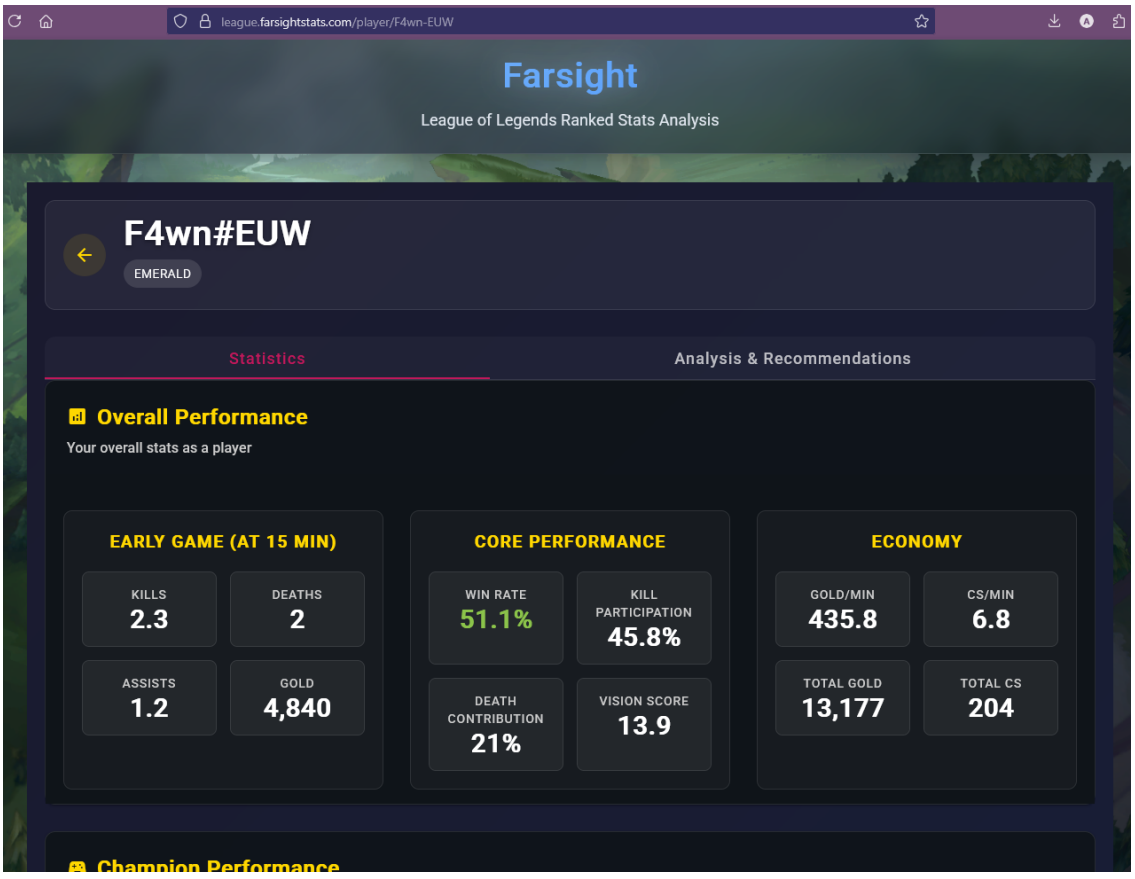


Figure 4.9: The results screen after a successful analysis.

Figure 4.10: Champion-specific data table.

The results screen (4.9) is divided in two tabs: Statistics and Recommendations. The Statistics tab presents player statistics through a card-based layout that groups related metrics logically, preventing information overload while maintaining visual hierarchy. Each of the cards can be hovered for detailed definitions, keeping a clean interface while providing contextual help on demand. Champion-specific statistics are presented in a straightforward table with clear, precise values, observable in 4.10. Color-coded performance indicators use green for strengths, blue for outstanding numbers, and red for areas that need improvement, leveraging color associations to instantly communicate performance levels.

The Recommendations tab, on the other hand, displays expandable/collapsible sections for each champion that a player consistently uses, in addition to the inherent "General Performance" one, as pictured below on 4.11. When expanded, these sections reveal concise and comprehensible feedback, allowing users to focus on specific recommendations, preventing information overload. Key usability decisions in both tabs include displaying all metrics with consistent decimal precision for technical accuracy, implementing responsive design for various screen sizes, and providing clear section headers that guide users through different aspects of their performance. The conclusions section uses actionable language with specific, constructive feedback rather than generic praise or criticism.

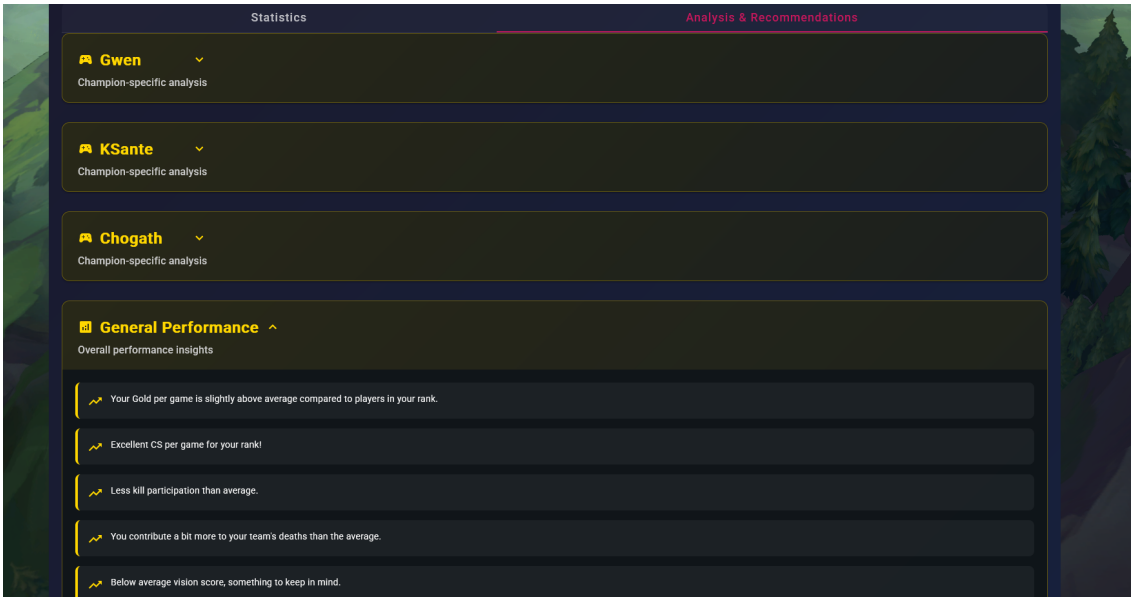


Figure 4.11: The Recommendations tab.

4.5.3 Flow

The application implements seamless navigation between screens using Angular Router, with URL-based access enabling direct linking to specific player results. The polling mechanism provides transparent progress updates during analysis, displaying queue position and estimated wait times to manage user expectations. This approach maintains user engagement during potentially long processing times while providing clear feedback about the status of the system.

4.5.4 Errors

Error handling is implemented at multiple levels, evidenced on 4.12: form validation prevents invalid submissions, network error handling provides clear error feedback and promotes app stability, and timeout management ensures users are not left waiting indefinitely. The interface prioritizes recoverability by allowing users to retry failed operations and providing clear paths back to the search interface.

Figure 4.12: The available user-friendly messages displayed by the error handling interface.

4.5.5 Iterations

The current interface is the result of several iterations based on user testing and feedback. Initially, conclusions were the main focus of the front page of the results screen, providing instant actionable recommendations, first on a general level, and then for each relevant champion. General feedback was that this was less appealing than an overview of statistics before moving to the conclusions: users were more interested in knowing numeric figures before being recommended improvement measures.

This was rectified, but the conclusions tab still needed refinement: users preferred champion recommendations to general ones. After a quick redesign to have champions at the top, players were then admitting to not noticing the General Performance section. As such, all sections in the "Analysis & Recommendations" tab were collapsed, except for General Performance, which is expanded by default, and user feedback was very positive regarding this arrangement.

Chapter 5

Evaluation

This chapter presents the empirical evaluation of whether game analytics facilitate player information processing and performance improvement. The evaluation employs survey-based methodology to collect quantitative and qualitative data from League of Legends players regarding their experience with analytics tools and their perceived impact on understanding gameplay patterns and consequent performance enhancement.

5.1 Survey Design

Evaluation of wether game analytics facilitate players' information processing can only be done through user input and perceived effectiveness of statistical data across a diverse population. Technical implementation may demonstrate successful identification of trends and patterns, but the only way to validate the present research hypothesis is by collecting systematic user feedback regarding the impact of analytics tools on understanding gameplay and acting on flaws.

A survey-based evaluation approach will be employed to capture both quantitative performance indicators and qualitative user experiences from active League of Legends players. This methodology enables the assessment of the effectiveness of analytics tools across different skill levels, experience ranges, and usage patterns while providing statistically analyzable data to support research conclusions.

To this end, the survey design will employ a structured questionnaire approach, divided in three sections, to garner context surrounding a participant, frame them in regard to their experience with the application and others with the same intent, and collect their qualitative and quantitative perception of the impact of the information at their disposal. This will be done in concise sections to minimize survey fatigue and incentivize high completion rates.

The first section will establish the respondent's familiarity with League of Legends and their in-game demographic (the region they play in), enabling meaningful stratification. Player experience will be captured through two year time range categories, from novice to veteran players, reliably categorizing users without incurring in overextending in the amount of possible answers.

Competitive ranking and region assessment will utilize the established League ranking system and region separation, respectively, enabling rank-stratified data and geographic segregation, for potential effectiveness mismatches across skill levels or player populations.

The second section will assess respondent's experience with the developed tool, as well as other game analytic platforms, to establish overall exposure and effectiveness. A detailed

impact assessment will take place, with a five-point Likert scale to quantify perceived performance improvements attributed to each tool, encompassing a wider range of possibilities and functionalities offered. Data obtained in this section will be useful in determining if different methodologies demonstrate varying levels of effectiveness in enabling critical thinking regarding gameplay.

The third and final section will be more focused and assertive, directly identifying the effectiveness of the application in facilitating player information processing and performance improvement.

Five-point Likert scales are once again employed, from "Not at all useful." to "Very useful", quantifying the help provided by the application. Behavioral impact measurement will assess wether players actually act upon their gameplay based on the analytics feedback, with responses ranging from active implementation to minimal changes, directly addressing wether analytics ease information processing.

Performance improvement assessment will capture player-perceived changes following usage of the analytics tool, categorizing responses from significant to not observable, direct evidence regarding enhancement after becoming aware of play patterns and trends.

The section will also include specific metric identification of which particular statistic contributed most significantly to their understanding and/or improvement, with options like win rate, champion-specific feedback, early game statistics, etc.

Lastly, two questions are placed regarding the continuing usage of the solution for pattern recognition and play enhancement, and directly asking if the tool is worth recommending to players seeking to improve, both with simple yes or no answers, to grasp the recognized value of the application to users.

5.1.1 Methodological Considerations

The survey design will implement several methodological safeguards to ensure data validity and reliability. Optional questions will be strategically employed to maximize completion rates while capturing valuable qualitative feedback from engaged respondents.

Anonymity will be maintained throughout the survey process to encourage honest responses regarding tool effectiveness and personal performance assessments. Response validation will be implemented through logical consistency checks and attention verification questions to ensure data quality, as well as answer randomization where applicable.

The survey will be deployed on a note in the solution itself, as well as active League of Legends discussion forums and communities, to maximize respondent reach and aim for diverse representation and more complete data.

Through this comprehensive survey design, the evaluation framework will provide robust evidence to address the central research question of whether game analytics facilitate player information processing and contribute to measurable performance improvement.

5.2 Survey Implementation, Iteration and Execution

The survey was deployed using Google Forms for maximum accessibility and to facilitate participation across multiple avenues. The Distribution channels chosen were communities dedicated to League of Legends with varying sizes, mainly on Reddit and Discord.

Reddit's defining characteristic is the subdivision of the website in circles dedicated to specific topics, and subreddit's like */r/leagueoflegends* (the main community devoted to League of Legends content with over 3.5 million members) and */r/summonerschool* (a League "subreddit dedicated to helping others learn and improve" *Summoner School* n.d., with 175 thousand subscribers) were ideal platforms for survey distribution and ample data points across answers.

Discord is also a community-building tool, focused on creating live-discussion groups with voice chat capabilities usually centered around specific topics. Servers dedicated to the game like the Riot Third Party Developer Community (gathering more than 21 thousand members) and the official community-ran League of Legends server (with over 400 thousand participants), as well as several smaller dimension servers of around 50 users with active League players, were handpicked and used to distribute the survey and broaden the reach of the feedback gathered.

The choice of distribution channels implies sampling targeted communities with diverse skill levels, from casual to competitive groups, from broad player representation to improvement-focused players, from large expansive communities to targeted, direct, and accessible groups.

While it is arguable that this approach introduces selection bias towards players actively engaged with online gaming circles, already receptive to the usage of analytics tools, this is precisely the primary target demographic for statistical-based feedback and improvement solutions, making the sample appropriate for evaluating such solutions' effectiveness among likely users. This does not affect the outcome of the study: receptibility to game analytics is not the main aspect being analyzed, but rather the usefulness and impact of such data.

Initial survey deployment included a dedicated section for evaluating Farsight specifically. However, pilot testing and early participants reported significant dropout when encountering questions focusing on a very recent solution. Tool-specific evaluation, particularly for the developed solution, appeared to create a barrier to survey completion and introduced severe selection bias towards participants more enthusiastic about using Farsight.

Furthermore, the prospect of having to return to the questionnaire for post-usage feedback was a contention point, hindering the number of responses obtained. As such, the survey was iterated upon, refined to focus not only on the implemented application but also established, popular analytic tools, including OP.GG, League of Graphs, Porofessor, Blitz.gg and Lolalytics. This proved successful in increasing participation rates while maintaining research validity by measuring the broader category of game analytics tools rather than focusing exclusively on a single implementation.

This refined survey also proved to be methodologically sound, as it addresses the core research question while avoiding practical limitations and user fatigue or lack of initiative with respect to unfamiliar tools. The decision to forsake Farsight-specific feedback represents a commitment to data quality and empirical evidence on whether game analytics facilitate information processing. The questions present in the final version can be observed in table 5.1, numbered according to their section.

Table 5.1: Survey questions.

Number	Question
Question-2.1	How long have you been playing League of Legends?
Question-2.2	What rank are you in League of Legends?
Question-2.3	What server do you usually play in?
Question-3.1	Have you used any websites/apps to help you track your statistics and improve?
Question-3.2	If you answered yes to the previous question, have they impacted your performance?
Question-4.1	Are the insights provided by the app(s) useful in helping you understand your play patterns?
Question-4.2	Did you ever act upon your gameplay according to the feedback that was provided?
Question-4.3	Have you ever noticed improvements in your performance after using the application(s)?
Question-4.4	What statistics helped you understand your improvement/lack thereof?
Question-4.5	Would you continue to use tools like Farsight or the previously listed ones to improve?
Question-4.6	Would you recommend the use of such tools to other players for improvement?
Question-4.7	Any additional feedback you'd like to share?

5.2.1 Data Collection Results

The survey achieved 71 total responses over a one month collection period. The response demographics demonstrate that most participants, nearly two-thirds (66.2%, or 47 users), have played League of Legends for more than seven years, with 15.5% reporting having played five to seven years and 12.7% from three to five, leaving only four participants below three-year activity. Figure 5.1 illustrates these results.

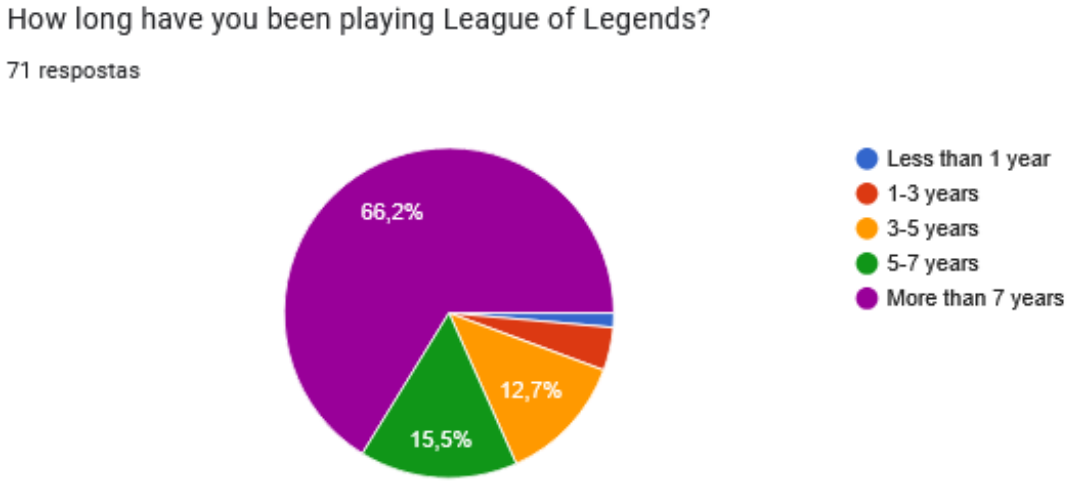


Figure 5.1: The resulting pie chart for League of Legends experience.

There was a very equal distribution of rank presence in upper echelons, with 16.9% for both Diamond and Master+, 21.1% for Emerald, and 19.7% belonging to both Platinum and Gold, once again with only four respondents below this level. These findings can be observed in figure 5.2.

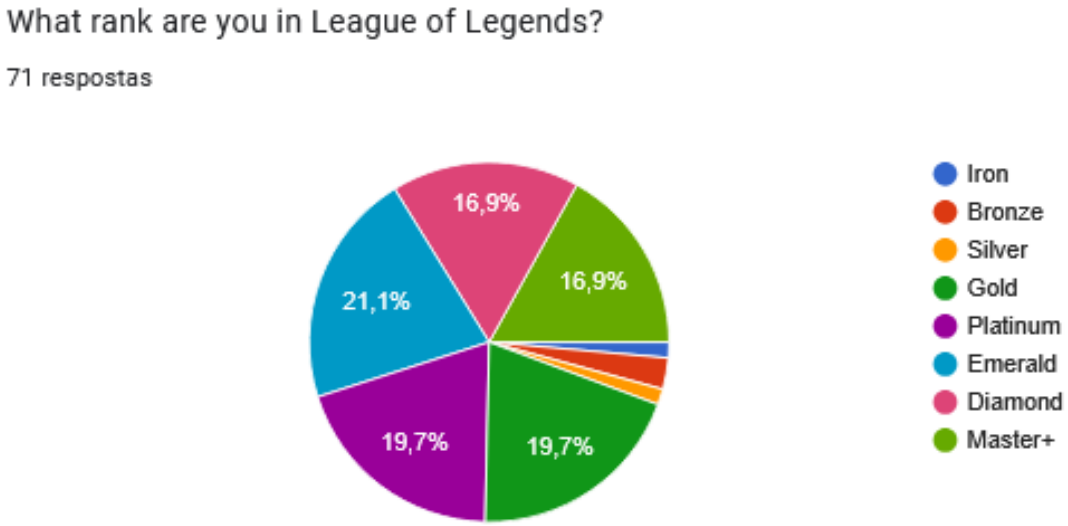


Figure 5.2: The graphic for Rank distribution across participants.

Server demographics were not so uniformly spread, with 54.9% of players reporting to play in the Europe West server, 23.9% on the North American one, and a fairly low amount for Europe Nordic & East, Oceania, Korea, Japan, Brazil, Turkey, Latin America North, Latin America South, and Southeast Asia. Other servers reported no users.

Regarding existing tools, a large majority of participants (62 out of 71) proclaimed having used statistics-tracking websites or apps for improvement purposes, with 9 denying interest in such tools, finishing their participation on this stage. Among the 62, only three reported to

have never used OP.GG, while other applications sat between 40-50% usage. On the Likert scale responses, where 1 means "Existing apps have not impacted my performance" and 5 "Existing apps have significantly impacted my performance", there was a fairly balanced division of the perceived impact: three was the most frequent answer, closely followed by four, one and five. Below is the resulting graphic on figure 5.3. In order of the mean scores, Porofessor has an average of 2.12 points, Blitz.gg of 2.41, LeagueOfGraphs sits at an average 2.91, OP.GG presents an average of 3.19 points, and Lolalytics exhibits a 3.43 average score.

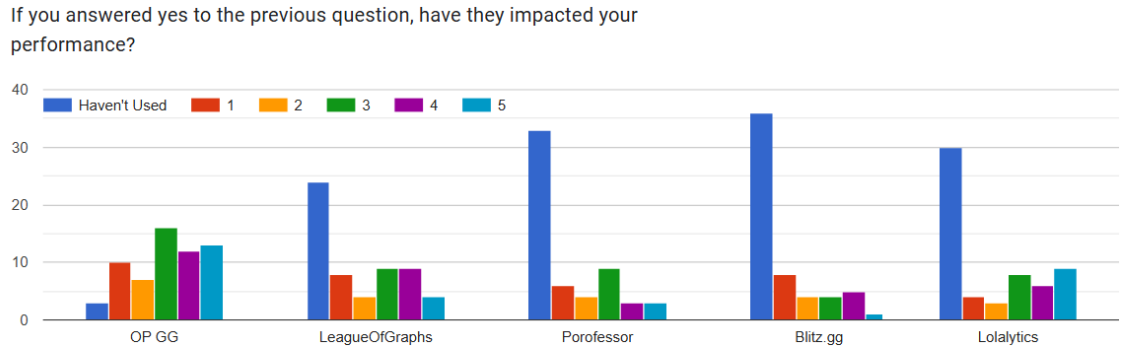


Figure 5.3: The resulting graphic for Existing Tools Impact assessment.

Regarding the adequacy of the existing tools, results tended to confirm the previous findings. When asked if the insights provided by the app proved useful in helping to understand play patterns, from 1: "Not at all" to 5: "Very useful", 28 (or 43.1%) of participants answered four, 11 went with three, and 10 for both two and five. Only six responded that the insights of existing apps had no impact at all. This distribution displays an average of 3.25 points. Figure 5.4 depicts this distribution.

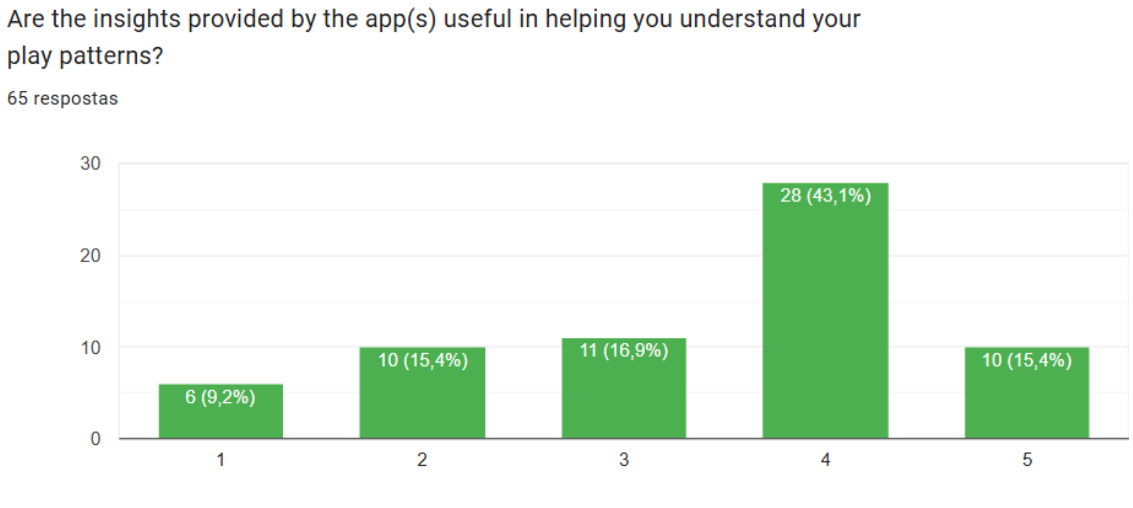


Figure 5.4: Results for the assessment of existing tool helpfulness on understanding play patterns.

As for the actual player initiative to address provided feedback, 33 of them reported to

"Partially" act upon statistical data in order to improve, while 19 "actively" implemented improvements on their gameplay and 13 made no attempts to apply the recommendations at their disposal, as can be observed in figure 5.5.

For the last mandatory input, on the perceived impact of analytics tools, a key concern of the study at hand, nearly three quarters of respondents (48 of them) consider to "have somewhat improved" after using these applications, with 9 observing "no improvements" and 8 "noticing significant improvements". Figure 5.6 illustrates these results.

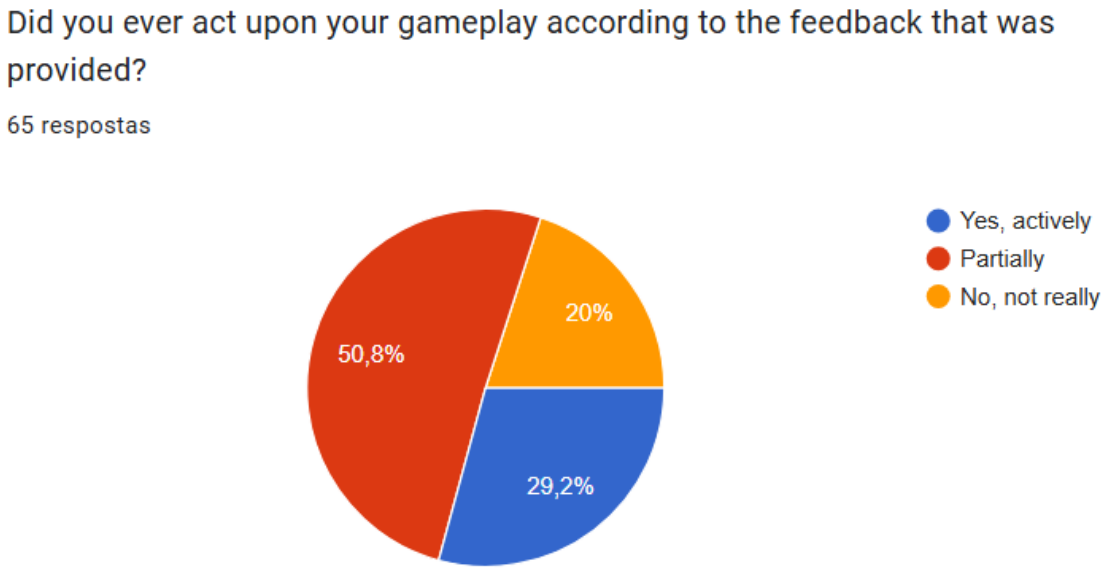


Figure 5.5: Players' initiative and motivation to act upon provided feedback.

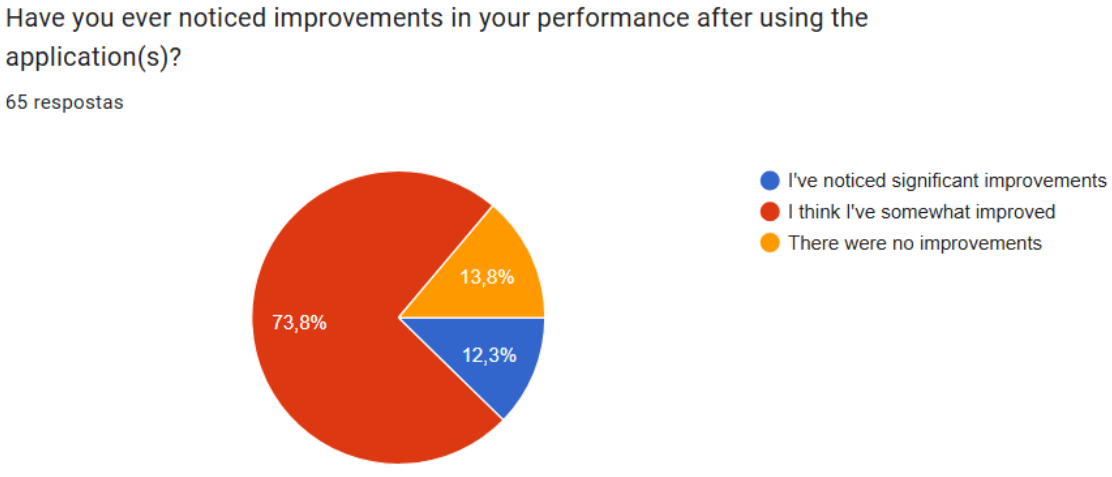


Figure 5.6: Perceived performance improvement after usage of analytics tools.

The concise nature of the survey elicited significant participation in optional inquiries. 49 in

57 participants admitted to continue using Farsight and similar tools for improvement, and 52 in 60 were keen to recommend such tools to other players for the same purpose.

Statistics such as creep score, vision score, win ratio and early game metrics were highly valued by players, as depicted in figure 5.7. 13 open-ended responses were also recorded, with insightful opinions, namely the hindrance behind live statistics, higher rank figures negatively influencing less capable players in lower ranks, the positive nature of the feedback for mistake-prone less informed players, and some general constructive criticism for Farsight.

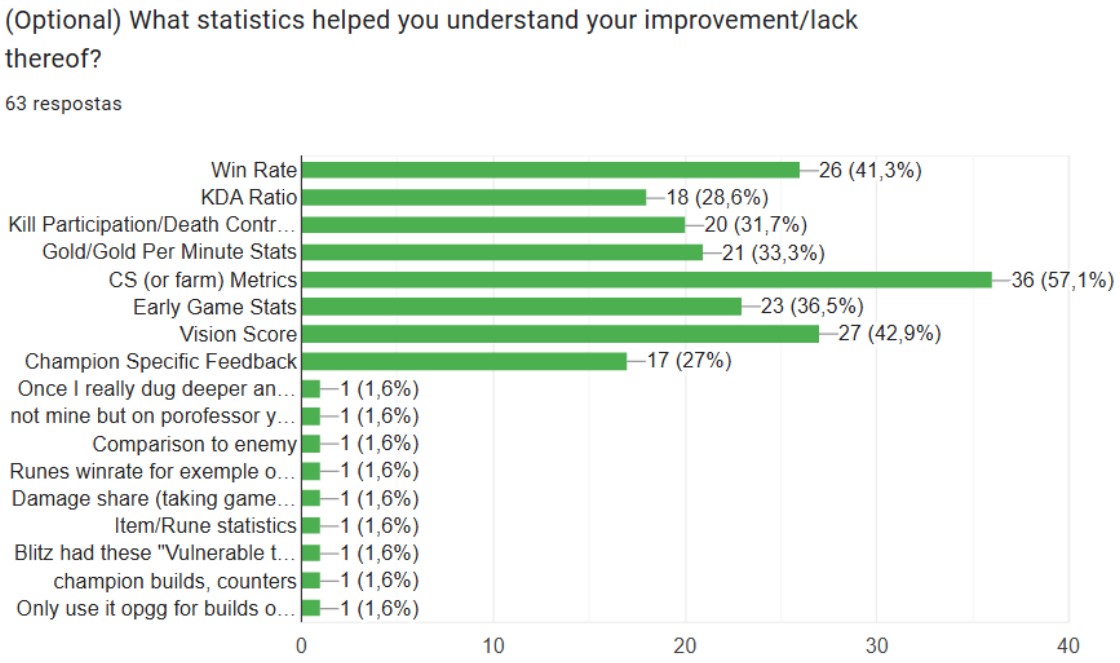


Figure 5.7: Player input on the most appropriate statistics in play pattern correction.

5.3 Discussion and Findings

5.3.1 Results Discussion

Building on the results presented above, this section discusses their implications and examines what the responses reveal about player behavior and the use of game analytics tools. The discussion is descriptive, framed by the central objective of the study of assessing how players perceive the value and impact of such tools, and whether these perceptions align with performance improvements.

In the demographics section, the responses show that the data set is primarily made up of veterans: nearly two-thirds have played for more than seven years and less than six percent report having played less than three years. The game itself is 16 years old, which means that 66.2% of all participants have played for at least half of its lifespan. Although this may be a consequence of bias in the data collection method, it is consistent with the communities used, which are frequented by more experienced players. Similarly, the rank distribution answers show higher survey participation among upper-ranked players. Once again, while this points to overrepresentation in this bracket, especially considering that competitors below

the Gold rank make up 51% of all ranked population (*League of Legends Rank Distribution in September 2025* 2025), players in the targeted communities are expected to be less casual, which can be reflected in their rank.

Overall, this skew toward experienced and higher-ranked participants provides useful context, but should be acknowledged as a slight limitation when interpreting the perceived impact of game analytics tools on performance. As the last demographic data point, server distribution is heavily weighted toward Europe West and North America, something that can be slightly correlated to player server distribution estimates in the western community, the main constituents of the channels where the survey was divulged, but limits eastern representation among the answers. (*League of Legends player count* 2025).

In terms of usage of game analytics applications, at a general level (before specific targeted feedback is collected), more than 87% of all participants report having used these tools to track statistics and improve performance. This suggests the popularity of such apps among survey responders, a strong inclination to seek detailed information on gameplay patterns and statistic trends.

Among the available options, OP.GG stands out as the most popular, followed by League Of Graphs and Lolalytics. These three solutions have the highest average perceived impact in the category, and point to a moderately positive effect of the usage of such tools in improving game execution. By contrast, Porofessor and Blitz.gg fewer users and lower perceived influence on improvements. The combination of these factors indicate a general predisposition among respondents to adopt game analytics tools, with OP.GG occupying a prominent role. Moreover, the perceived usefulness of such applications appears to be positive overall, even if the degree of impact varies depending on the specific tool.

While the previous section offered a broad overview of the participants' notion of the applications in the domain, the adequacy section aims to go in-depth and assess the practicality of these solutions and their effect. The first question directly addresses the usefulness of game analytics tools in helping understand play patterns. The responses are strongly positive, with an observed median of four in the Likert scale, demonstrating a very positive perception of the tools in communicating additional game data to players in a comprehensible manner.

Regarding the initiative toward acting on the feedback provided by the solutions, 80% of players report to have acted on their gameplay, even if partially, after utilizing statistics tracking applications. This once again denotes a significant effect on play trends among users. In the last mandatory input of the survey, players are asked to explicitly state their perceived impact on performance improvements after using game analytics tools. A similar number of respondents sit at the extremes, but the vast majority reports to notice at least some improvements. This data, combined with the fact that only 13.8% of users noticed no improvements, suggests a productive observed influence of these applications on performance enhancement, and strongly points to their beneficial nature towards information interpretation and actionability in gameplay corrections.

Among the optional questions, 86% of 57 participants share a willingness to continue using game analytics tools with the intent of improving, pointing to a preconceived positive notion around them. This is further emphasized by the fact that 52 in 60 respondents report a keenness to recommend such tools to other players as a means of promoting improvement. Finally, the open-ended answers indicate that players in the data sample believe that critical analysis of the available information is a beneficial inclusion in day-to-day gameplay, as long as its contextualized.

5.3.2 Findings Overview

The collective findings from this survey reveal several key insights about the relationship between game analytics tools and player behavior in League of Legends. First, there exists a strong correlation between player experience and analytics tool adoption, with veteran players (those with 7+ years of experience) demonstrating both higher usage rates and more positive perceptions of tool effectiveness. This suggests that experienced players possess the contextual knowledge necessary to interpret and act on statistical feedback in a meaningful way.

Second, the data demonstrates a clear progression from tool usage to behavioral change to perceived performance improvement. With 87% of participants reporting analytics tool usage, 80% indicating they have modified their gameplay based on feedback, and 86.2% observing at least some performance improvements, the findings suggest a functional pathway from data consumption to performance enhancement. This progression supports the theoretical proposition that game analytics can facilitate information processing and translate complex gameplay data into actionable insights.

Third, the widespread adoption and continued usage intentions among participants underscore the practical value players derive from analytics tools. With 86% expressing willingness to continue using such tools and 87% willing to recommend them to others, the data reveals that players not only find these tools beneficial but actively endorse their use within their communities. This sustained engagement and advocacy behavior indicates that game analytics tools have successfully established themselves as valuable resources for performance improvement rather than mere novelties.

Finally, the overwhelmingly positive response to questions about understanding play patterns and the high rate of gameplay modification based on analytics feedback indicate that these tools successfully bridge the gap between raw statistical data and player comprehension. This finding directly addresses the central research question regarding game analytics' role in facilitating information processing and suggests that well-designed analytics tools and adequately contextualized insights can indeed enhance players' ability to interpret and act upon performance data.

5.3.3 Implications

These findings contribute to the understanding of how players process complex gaming information. The high correlation between analytics tool usage and perceived improvement validates theories of data-driven learning in competitive game environments. The results suggest that when statistical feedback is properly contextualized, players can effectively translate numerical data into strategic behavioral changes, supporting cognitive load theory in gaming contexts.

For game developers, these findings indicate that integrating analytics features directly into games could enhance player engagement and skill development. The emphasis on contextualized feedback suggests that raw statistics alone are insufficient - analytics tools must supply additional data gathering and refinement, and provide interpretive frameworks that help players understand what the numbers mean for their specific gameplay situations and circumstances.

For game analytics tool developers, the results highlight the importance of user experience in determining tool effectiveness. The willingness to recommend tools to others suggests that successful analytics platforms create positive feedback loops within player communities.

The study demonstrates that analytics tools can serve as effective coaching supplements for players across different experience levels, provided the tools offer suitable contextualization and user-friendly implementation. Highly experienced players possess the contextual knowledge to interpret feedback meaningfully, and the tools should adapt to more casual players to convey recommendations clearly. The high adoption rates and positive user experiences indicate that analytics tools have successfully bridged the gap between complex statistical data and actionable player insights, validating the theoretical framework underlying this research.

Chapter 6

Conclusions

This chapter presents an overview of the conclusions drawn from the work undertaken, in an effort to assess the impact of game analytics on player's information processing and consequent improvement. The findings are examined in relation to the initial research objectives, followed by an acknowledgment of the study's limitations. A discussion of potential improvements is then presented, outlining to extend the research contributions in game analytics and player performance enhancement.

6.1 Research Objectives

This dissertation set out to address the Research Question: "How does one extract and analyse information from a game in order to positively impact a player's performance and experience?". This question addresses a fundamental challenge in game analytics: transforming complex, hidden statistical data into meaningful insights that players can understand and act upon.

The work developed in this thesis tackled the research objective by targeting specifically one game, League of Legends, conducting research to understand how to extract data from it, and building a data extraction, statistics analysis tool capable of interpreting data and generating actionable recommendations for users.

By defining statistical thresholds, the tool critically compares a subject's numbers with players in similar conditions, and presents contextualized feedback to aid in processing the information extracted and identifying strengths and potential improvement areas. Furthermore, it leverages similar existing applications, with different methodologies but the same purpose of exposing game analytics, to assess the adequacy of this technique.

A survey was conducted to validate this approach, with positive outcomes: the results obtained show that solutions that collect previously hidden statistics and expose them to users, like the one developed through this thesis, are very popular, and the overall sentiment is that they constitute a noticeable impact on player experience and an effective complement to gameplay as a means to identify patterns and improve. In this regard, the fundamental objective of the thesis has been achieved successfully. Game data was extracted, interpreted, contextualized, and adequately conveyed to players, who recognize value in this information and report positive changes in their experience.

This method is not exclusive to League of Legends; the methodology used can be adapted to other games where some of the information is technically publicly available but not directly accessible or presented within the game interface. Moreover, the approach taken does not represent an imperative, definitive answer to the research question. It proved to be

viable and adequate, but represents one of multiple possible approaches to game analytics implementation, with potential for further refinement and adaptation to different gaming contexts and player needs.

This research contributes to the understanding of how game analytics can bridge the gap between raw statistical data and actionable player insights, providing a foundation for future developments in player performance enhancement tools.

6.2 Study Limitations

Although this research successfully demonstrates the viability of game analytics to facilitate player information processing, select limitations must be acknowledged. First, the evaluation was conducted exclusively through user surveys, which rely on self-reported perceptions rather than objective performance measurements. While participants reported perceived improvements, the study did not measure actual gameplay performance changes through match statistics or rank progression data.

Second, the survey sample exhibited demographic bias toward experienced and higher-ranked players, with nearly two-thirds having played for over seven years and underrepresentation of players in the bottom 50% of ranks. This skew limits the generalizability of findings to the broader League of Legends player base, particularly casual players.

Third, the technical implementation focused solely on League of Legends, utilizing its specific API structure and game mechanics. The methodology is theoretically adaptable to other games, but the practical constraints of different API availability and structure, data formats, and game-specific metrics were not empirically tested.

Finally, the study's timeframe limited the scope of long-term impact assessment. The survey captured immediate user reactions and short-term behavioral changes, but did not examine sustained usage patterns or long-term performance improvements that might result from continued analytics tool usage.

These limitations suggest directions for future research involving objective performance measurement, broader demographic sampling, cross-game validation, and longitudinal studies.

6.3 Future Improvements

As Farsight gathers more users and starts observing returning players, evaluation methodology can evolve from a Google Forms survey to a simple questionnaire imbued in the website itself, increasing accessibility, feedback reach, and potentiating the return of application-specific critiques (previously removed as referred in the Survey Iteration process).

Graphical representation of the resulting metrics was foregone when committing to single player current status analysis rather than dashboard-oriented continuously updating feedback. Early user opinion also prioritized raw data acknowledgment rather than potentially confusing visual depictions of gameplay trends. However, future updates could include aids like graphs or charts to help less statistically proficient users accommodate to the site and promote higher usage numbers.

As it stands, given the current lightweight single-user operation, the chosen implementation is adequate for the available development API key, resulting in a properly engineered application that adapts to the external limitations encountered while still providing a smooth

experience for users. However, in a future with more users and an approved production key, the backend process could be improved using concurrent threads, a rate limiter and an orchestrator to synchronize several requests, dynamically update delays, and potentially set up priority analysis and better monitoring.

Adding timing details fields to the Statistics, identifying characteristics such as game patch or competitive season, and saving it with the data could prove to be beneficial added context for the app's conclusions and consequent actions taken by the player to improve. The adopted design allows this iteration to be easily implemented.

Support players noticed a skew towards gold and creep relevance in the final recommendations provided. These are not the most useful in their particular case, as their role aims to provide high value with low resources. A weighted average system to address this skew was already in advanced development, and in the future, position-based feedback can mitigate this.

Bibliography

Association for Computing Machinery (2023). url: https://www.acm.org/code-of-ethics (visited on 12/29/2023).

Cahn, Aaron et al. (2016). "An empirical study of web cookies". In: *Proceedings of the 25th international conference on world wide web*, pp. 891–901.

Chu, Xu et al. (2016). "Data cleaning: Overview and emerging challenges". In: *Proceedings of the 2016 international conference on management of data*, pp. 2201–2206.

CS2 Active Players (2024). url: https://steamdb.info/app/730/charts/ (visited on 01/05/2024).

DeCoster, Jamie (2001). "Transforming and restructuring data". In: *University of Alabama, Department of Psychology, 7* 8.

Developer Portal (n.d.). url: https://developer.riotgames.com/docs/portal#web-apis_api-keys.

Eckhardt, Jonas, Andreas Vogelsang, and Daniel Méndez Fernández (2016). "Are "Non-functional" Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice". In: *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 832–842. doi: 10.1145/2884781.2884788.

Fayyad, U. M. (1996). "Advances in Knowledge Discovery and Data Mining". In: *AAAI Press*.

Fernandes, Lucas V., Carla D. Castanho, and Ricardo P. Jacobi (2018). "A Survey on Game Analytics in Massive Multiplayer Online Games". In: *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pp. 21–2109. doi: 10.1109/SBGAMES.2018.00012.

Fuentes, Carolina et al. (Aug. 2017). "A systematic literature review about technologies for self-reporting emotional information". In: *Journal of Ambient Intelligence and Humanized Computing* 8, pp. 593–606. doi: 10.1007/s12652-016-0430-z.

Getting Cloudflare Tunnels to connect to the Cloudflare Network with QUIC (2021). url: https://blog.cloudflare.com/getting-cloudflare-tunnels-to-connect-to-the-cloudflare-network-with-quic/ (visited on 10/20/2021).

Gudivada, Venkat N (2017). "Data analytics: fundamentals". In: *Data analytics for intelligent transportation systems*. Elsevier, pp. 31–67.

Importance of data collection in healthcare (2023). url: https://www.foreseemed.com/importance-of-data-collection-in-healthcare (visited on 12/30/2023).

Junior, Jailson and Cláudio Campelo (Dec. 2023). "League of Legends: Real-Time Result Prediction". In: pp. 1–8. doi: 10.21528/CBIC2023-161.

Kebede, Rahel, Annika Moscati, and Peter Johansson (Aug. 2020). "Semantic Web and Linked Data for Information Exchange between the Building and Product Manufacturing Industries: A Literature Review". In: pp. 248–265. doi: 10.46421/2706-6568.37.2020.paper018.

Kumar, Vaibhav and ML Garg (2018). "Predictive analytics: a review of trends and techniques". In: *International Journal of Computer Applications* 182.1, pp. 31–37.

LeagueOf (2023). *League Of Graphs*. url: https://www.leagueofgraphs.com/ (visited on 12/31/2023).

League of Legends player count (2025). url: https://www.thespike.gg/league-of-legends/beginner-guides/league-of-legends-player-count.

League of Legends Rank Distribution in September 2025 (2025). url: https://www.esportstales.com/league-of-legends/rank-distribution-percentage-of-players-by-tier.

Mafra, Priscilla et al. (Aug. 2016). "Towards Guidelines for Preventing Critical Requirements Engineering Problems". In: *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, pp. 25–29. doi: 10.1109/seaa.2016.50. url: http://dx.doi.org/10.1109/SEAA.2016.50.

Manikandan, S (2010). "Data transformation". In: *Journal of Pharmacology and Pharmacotherapeutics* 1.2, p. 126.

Maymin, Philip Z. (2021). In: *Journal of Quantitative Analysis in Sports* 17.1, pp. 11–27. doi: doi:10.1515/jqas-2019-0096. url: https://doi.org/10.1515/jqas-2019-0096.

Moreira, João, Andre Carvalho, and Tomás Horvath (2018). *A general introduction to data analytics*. John Wiley & Sons.

El-Nasr, Magy Seif, Anders Drachen, and Alessandro Canossa (2016). *Game analytics*. Springer.

Oates, Briony J, Marie Griffiths, and Rachel McLean (2022). *Researching information systems and computing*. Sage.

Ong, Hao Yi, Sunil Deolalikar, and Mark Peng (Mar. 2015). "Player Behavior and Optimal Team Composition for Online Multiplayer Games". In.

Oviatt, Sharon (2006). "Human-centered design meets cognitive load theory: designing interfaces that help people think". In: *Proceedings of the 14th ACM International Conference on Multimedia*. MM '06. Santa Barbara, CA, USA: Association for Computing Machinery, pp. 871–880. isbn: 1595934472. doi: 10.1145/1180639.1180831. url: https://doi.org/10.1145/1180639.1180831.

Peffers, Ken et al. (Jan. 2007). "A design science research methodology for information systems research". In: *Journal of Management Information Systems* 24, pp. 45–77.

Player Universally Unique IDentifiers and a New Security Layer (2018). url: https://www.riotgames.com/en/DevRel/player-universally-unique-identifiers-and-a-new-security-layer (visited on 11/08/2018).

PUUIDs and Other IDs (2019). url: https://riot-api-libraries.readthedocs.io/en/latest/ids.html.

Rehman, Ikhlaq ur (2019). "Facebook-Cambridge Analytica data harvesting: What you need to know". In: *Library Philosophy and Practice*, pp. 1–11.

Research, BCC (2023). *Video games: Global markets*. url: https://www.bccresearch.com/market-research/information-technology/video-game-market.html.

Runkler, Thomas A (2020). *Data analytics*. Springer.

Samadhiya, Durgesh, Su-Hua Wang, and Dengjie Chen (2010). "Quality models: Role and value in software engineering". In: *2010 2nd International Conference on Software Technology and Engineering*. Vol. 1. IEEE, pp. V1–320.

Su, Yanhui, Per Backlund, and Henrik Engström (2021). "Comprehensive review and classification of game analytics". In: *Service Oriented Computing and Applications* 15, pp. 141–156.

Summoner School (n.d.). url: https://www.reddit.com/r/summonerschool/.

The Importance and Benefits of Data Collection in Healthcare (2023). url: https://obi.services/blogs/the-importance-of-data-collection-in-healthcare-and-its-benefits/ (visited on 12/30/2023).

What is a CDN? (N.d.). url: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/.

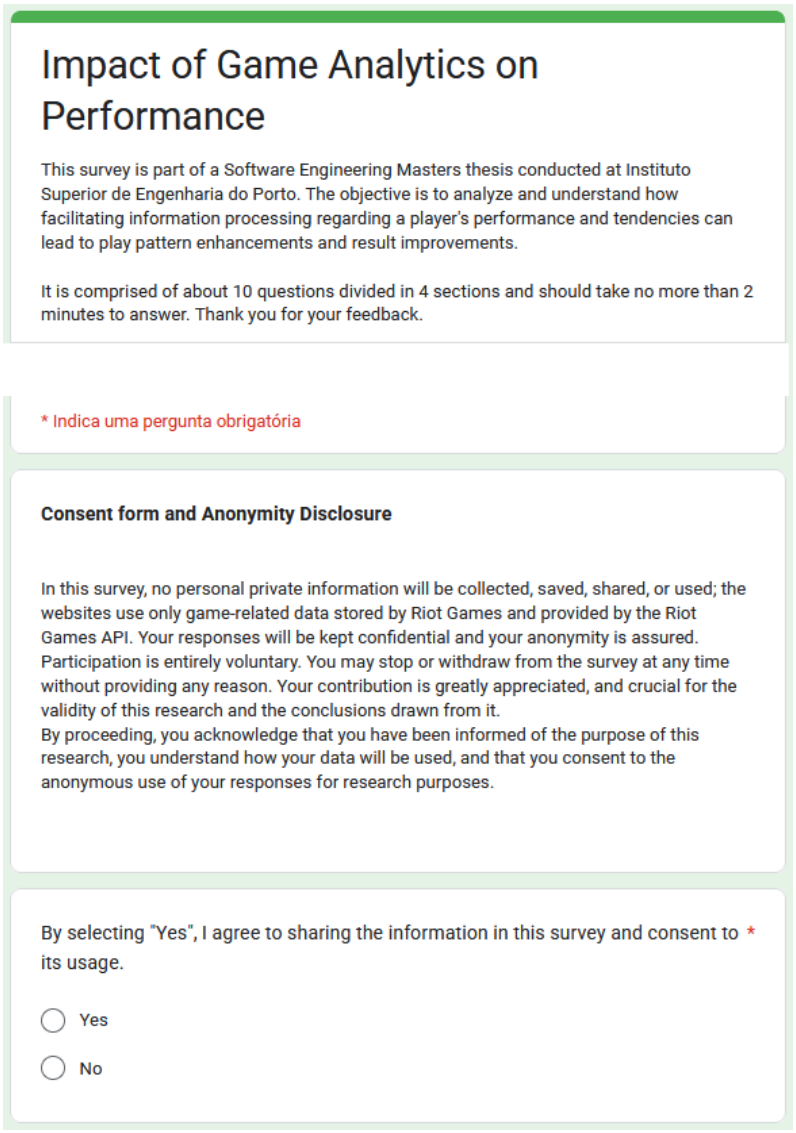
What is a PICOC? (2023). url: https://cebma.org/resources/frequently-asked-questions/what-is-a-picoc/ (visited on 12/30/2023).

What is Data Collection? Why is it Important for Your Business? (2023). url: https://emeritus.org/blog/data-analytics-what-data-collection/ (visited on 12/30/2023).

Appendix A

Survey

The structure of the final version of the survey, elaborated in order to evaluate the main question of the thesis and assess data tracking applications' impact, is shown in figures A.1, A.2, A.3, A.4, and A.5.



Impact of Game Analytics on Performance

This survey is part of a Software Engineering Masters thesis conducted at Instituto Superior de Engenharia do Porto. The objective is to analyze and understand how facilitating information processing regarding a player's performance and tendencies can lead to play pattern enhancements and result improvements.

It is comprised of about 10 questions divided in 4 sections and should take no more than 2 minutes to answer. Thank you for your feedback.

* Indica uma pergunta obrigatória

Consent form and Anonymity Disclosure

In this survey, no personal private information will be collected, saved, shared, or used; the websites use only game-related data stored by Riot Games and provided by the Riot Games API. Your responses will be kept confidential and your anonymity is assured. Participation is entirely voluntary. You may stop or withdraw from the survey at any time without providing any reason. Your contribution is greatly appreciated, and crucial for the validity of this research and the conclusions drawn from it.
By proceeding, you acknowledge that you have been informed of the purpose of this research, you understand how your data will be used, and that you consent to the anonymous use of your responses for research purposes.

By selecting "Yes", I agree to sharing the information in this survey and consent to its usage. *

- ◯ Yes
- ◯ No

Figure A.1: First page of the questionnaire, focused on consent information and anonymity disclosure.

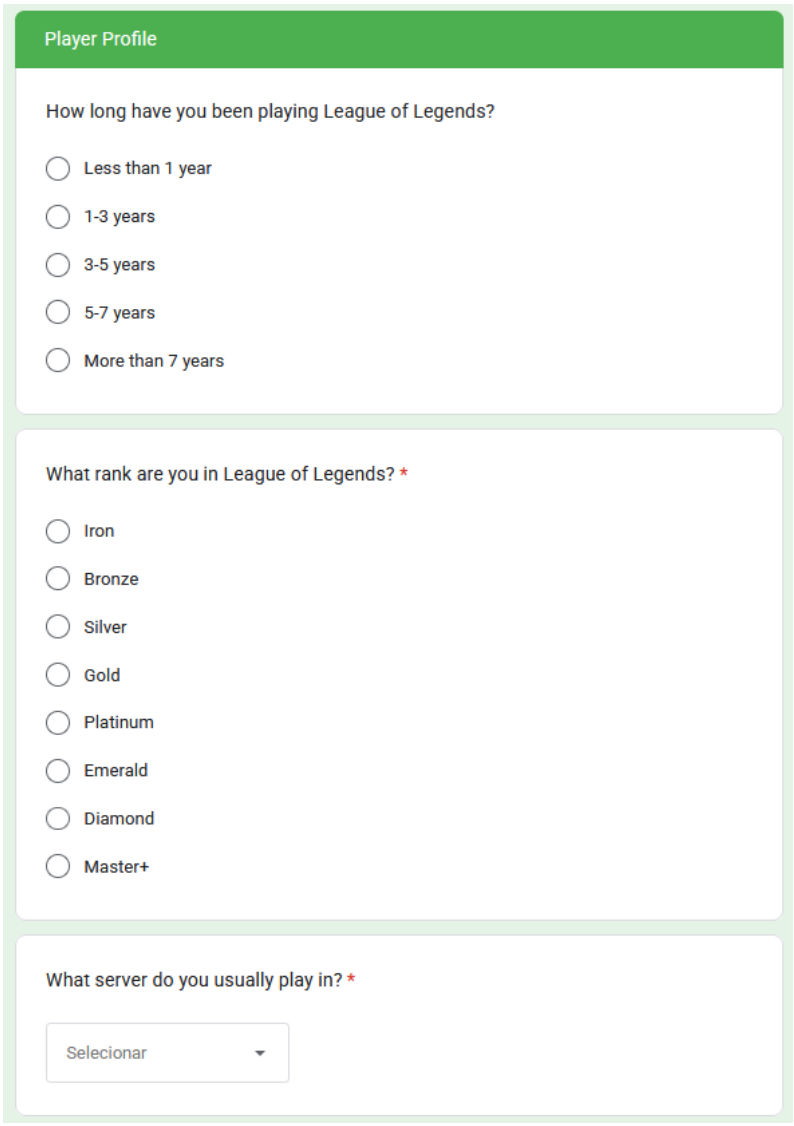
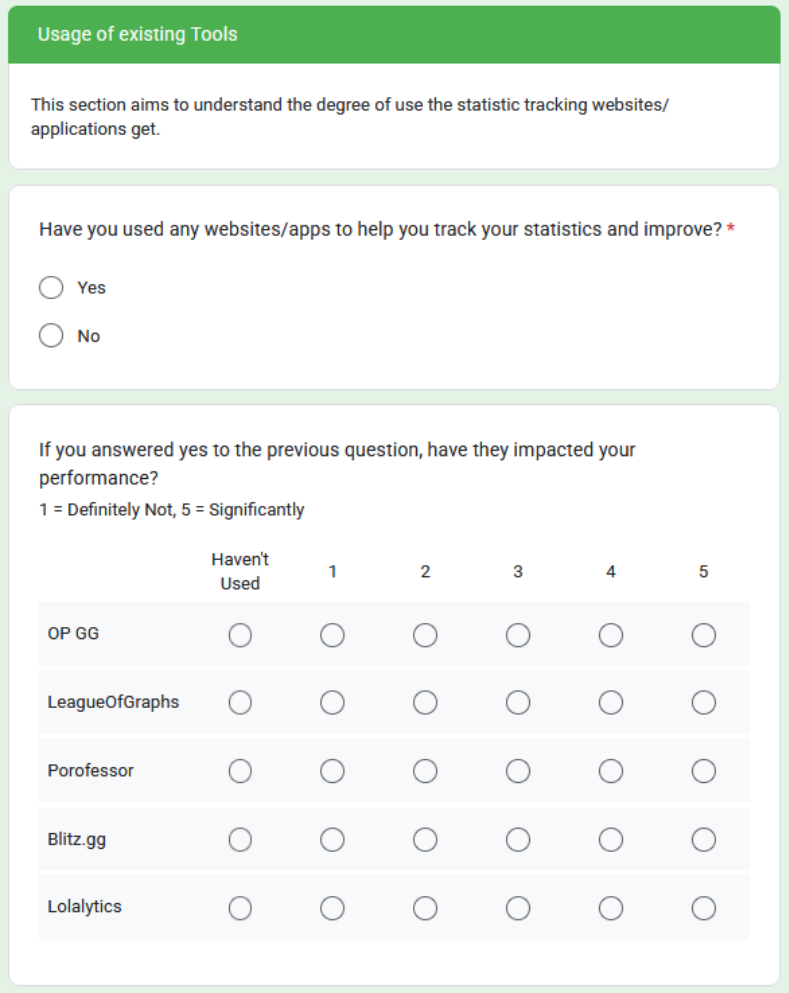


Figure A.2: Second page of the questionnaire, focused on profiling the player.



Usage of existing Tools

This section aims to understand the degree of use the statistic tracking websites/ applications get.

Have you used any websites/apps to help you track your statistics and improve? *

- Yes
- No

If you answered yes to the previous question, have they impacted your performance?

1 = Definitely Not, 5 = Significantly

	Haven't Used	1	2	3	4	5
OP GG						
LeagueOfGraphs						
Porofessor						
Blitz.gg						
Lolalytics						

Figure A.3: Third page of the questionnaire, assessing the usage of existing game analytics tools.

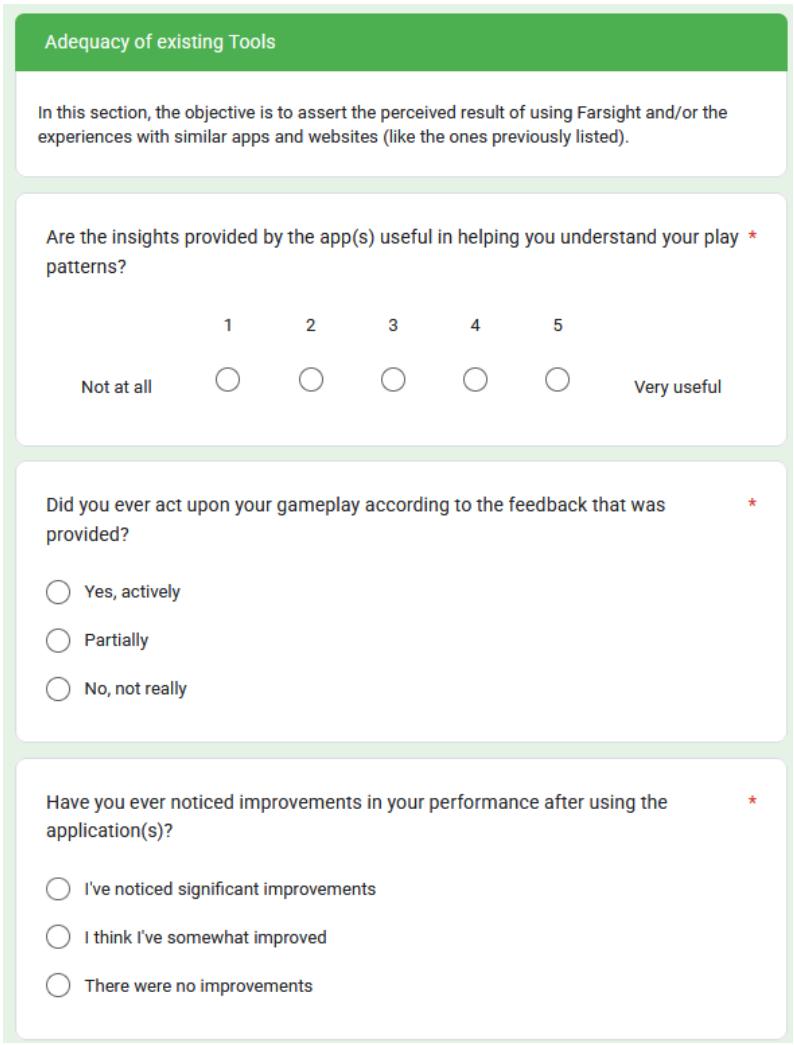


Figure A.4: First half of the fourth page of the questionnaire, addressing the adequacy of game analytics tools and their perceived impact.

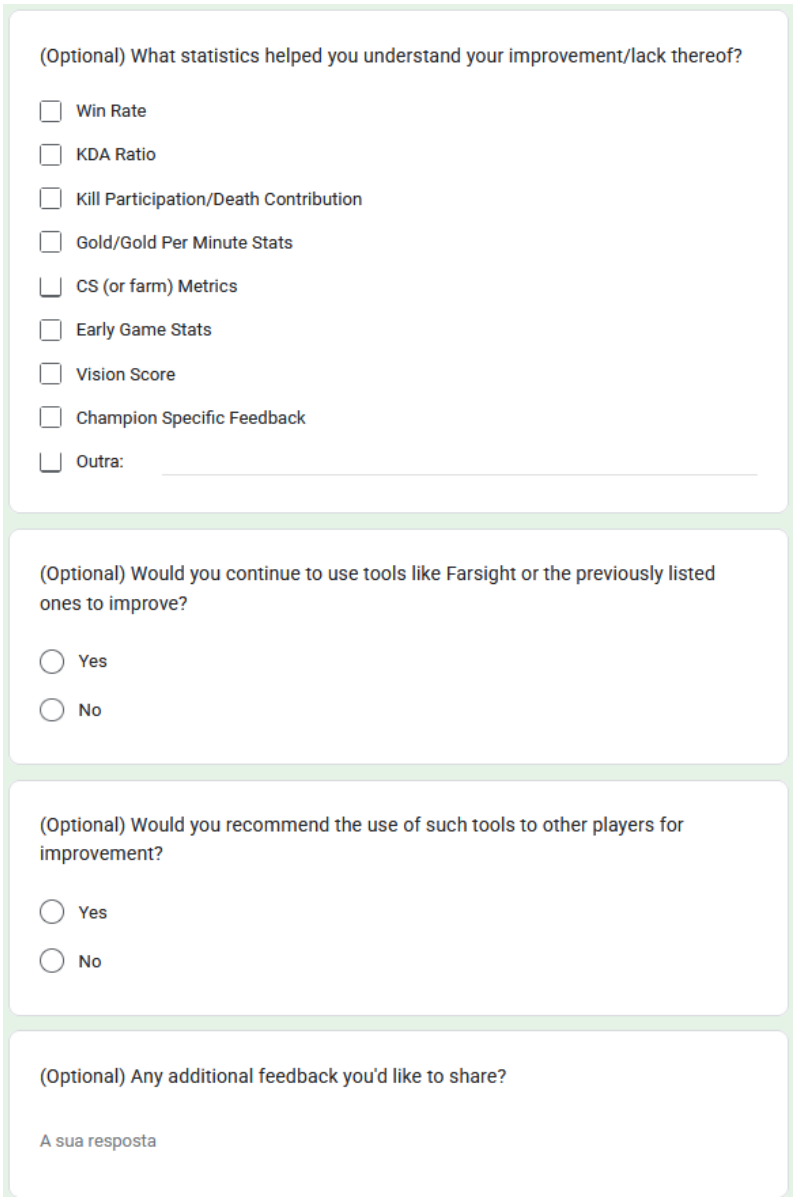


Figure A.5: First half of the fourth page of the questionnaire, addressing the adequacy of game analytics tools and their perceived impact.