

# Ferramenta de auxílio a jogadores de League of Legends

**Matheus Avi, Ricardo de la Rocha Ladeira (orientador)**

Instituto Federal Catarinense – Campus Blumenau  
Rua Bernardino José de Oliveira, nº 81 – Badenfurt – Blumenau/SC – Brasil  
{matheusavi.47@gmail.com, ricardo.ladeira@ifc.edu.br}

## 1. Introdução

Este documento versa sobre o desenvolvimento de uma ferramenta para ajudar jogadores do jogo League of Legends, produzido pela Riot Games.

De acordo com o *website* da produtora do jogo, League of Legends é um jogo de equipe e estratégia que tem como objetivo principal destruir uma construção da base inimiga, conhecida como Nexus (Figura 1) (RIOT GAMES, 2018).



**Figura 1. Nexus azul.<sup>1</sup>**

No início da partida, cada um dos dez jogadores, que formam dois times de cinco, escolhe um personagem para controlar. Existem aproximadamente 148 personagens divididos em seis principais papéis (RIOT GAMES, 2018), sendo eles:

1. Assassino, que entra na luta, elimina seu alvo rapidamente e sai;

---

<sup>1</sup> Disponível em:

[https://vignette.wikia.nocookie.net/leagueoflegends/images/f/fd/Nexus\\_azul.png/revision/latest?cb=20141122125741&format=original&path-prefix=pt-br](https://vignette.wikia.nocookie.net/leagueoflegends/images/f/fd/Nexus_azul.png/revision/latest?cb=20141122125741&format=original&path-prefix=pt-br)

2. Tanque, que tem grande resistência e protege os campeões<sup>2</sup> mais fracos;
3. Lutador, que é um meio termo entre os dois, tem defesa e dano médio;
4. Mago, que possui um tipo diferente de dano, o dano mágico. Possui um bom controle de grupo, ou seja, é capaz de atrapalhar vários adversários ao mesmo tempo. É frágil;
5. Atirador, que causa dano físico a distância e também é frágil;
6. Suporte, que possui habilidades para auxiliar os aliados ou atrapalhar os inimigos;

No decorrer da partida, cada jogador evolui de forma diferente de acordo com seus feitos, tornando-se cada vez mais forte. Essa combinação de diferentes personagens, jogadores, funções e evolução tornam cada partida única. Além disso, dentro do jogo existe uma classificação, conhecida como *elo*, que serve para fazer com que os jogadores meçam sua evolução competindo entre si.

De acordo com Aqours (2018), são 100 milhões de jogadores ativos por mês, além disso, o jogo movimenta uma grande quantia de dinheiro e possui um grande público em seus campeonatos. O MSI 2017 (Mid-Season Invitational ou Campeonato Internacional de Meia Temporada), que ocorreu no Brasil, contou com 364 milhões de espectadores e uma premiação de 1,7 milhões de dólares. O campeonato mundial de 2017, que ocorreu na China, chegou a contar 80 milhões de espectadores durante as semifinais e teve a premiação de 4 milhões de dólares, sendo que 2,696 milhões de dólares vieram da contribuição de fãs.

A maior motivação para o trabalho é a de auxiliar a comunidade do jogo a se fortalecer, fornecendo informações concretas aos jogadores, de maneira inédita, diferentemente das ferramentas existentes, citadas na Seção 2. (“Trabalhos correlatos existentes”).

Tornando essas informações mais acessíveis, os jogadores iniciantes podem melhorar seu desempenho no jogo, aumentando suas chances de vencer a partida, o que deixa o jogo mais amigável aos olhos de novos jogadores e motivador aos experientes.

### 1.1. Tema/Problema

Conforme Golart, Kroeff e Gavillon (2017), existe uma variedade incalculável de ações que podem ser tomadas por um jogador em cada partida, fazendo com que grande parte do aprendizado venha da experiência do jogador. Este conjunto de ações somado a vários fatores do jogo (grande número de campeões com mecânicas diferentes e constantes alterações no mapa) torna o aprendizado difícil e moroso para o iniciante, pois ele não tem experiência suficiente para saber quais são as melhores ações.

De acordo com Seula (2017), jogos competitivos como o League of Legends têm como alvo um público mais engajado, fazendo com que os jogadores sempre procurem jogar melhor, e uma curva de aprendizado mais longa, que torna a melhora da classificação dos jogadores demorada. Ainda de acordo com Seula (2017), como o jogo passa por frequentes mudanças, mantém os jogadores motivados por longo prazo e em busca de melhorias, pois quem não se atualiza junto com o jogo, consequentemente fica em desvantagem pela falta de conhecimento. Também conclui-se que mesmo jogadores

---

<sup>2</sup> Personagem controlado por um jogador.

experientes, caso parem de jogar por algum tempo, podem apresentar dificuldades ao escolher as ações a serem tomadas.

## **1.2. Objetivos Propostos/Solução dos Problemas**

O objetivo do trabalho é construir uma ferramenta que auxilie jogadores exibindo informações que podem ser úteis durante a partida, como:

- Chance de vencer a partida;
- Itens que deve-se comprar;
- Com que oponente o jogador deve tomar cuidado;
- Dicas de ações a serem tomadas;
- Jogadores confiáveis.

O trabalho visa a entrega de uma melhor experiência ao jogador, fornecendo as informações descritas acima por meio de uma ferramenta *web*, sendo assim um atalho para o jogador não precisar passar por várias experiências com erro para realizar o seu aprendizado. A partir destas informações, o jogador poderá tomar melhores decisões e com o tempo melhorar seu modo de jogar, tornando assim sua experiência no jogo mais agradável e tendo a oportunidade de subir seu *elo*.

## **1.3. Escopo**

Este trabalho trata somente de um dos modos do jogo, chamado “Fila Solo/Duo”, que é o principal foco do jogo. Neste modo, antes da partida começar, são definidos os campeões que os jogadores vão controlar (sendo cinco contra cinco, cada um controlado por um jogador) e a posição (*lane*) que cada um vai ocupar durante a partida.

O projeto consiste em uma aplicação *web* e responsiva acessível a partir de qualquer dispositivo com um navegador *web*, dividida em três partes. Na página inicial deverão ser inseridos nome de usuário e região. Após isso o usuário deve selecionar se deseja ver os dados da partida atual ou o histórico de partidas (descritos a seguir). No histórico de partidas será possível visualizar partidas que o jogador já jogou e detalhes delas. Na página da partida atual o jogador contará com vários dados referentes a partida atual, tanto informativos quanto para auxiliá-lo a tomar decisões na partida que ele está jogando (não fornecendo exatamente qual ação tomar, pois como a partida é imprevisível, não é possível oferecer informações exatas).

A ferramenta não será integrada ao jogo diretamente, somente à *API* (*Application Programming Interface*) da Riot Games, ou seja, será necessário ter um celular, uma segunda tela ou alternar entre a aplicação e o jogo para que os dados sejam visualizados durante a partida.

## **1.4. Viabilidade do Projeto**

O projeto é viável economicamente, já que todas as ferramentas utilizadas para desenvolvimento são livres ou de custo zero para desenvolvedores independentes.

Sobre a viabilidade técnica, a hospedagem para teste pode ser feita em qualquer servidor linux que atenda aos requisitos definidos em 3.1.2.

Sobre a viabilidade operacional, o usuário tem três maneiras de utilizar o sistema, sendo elas:

1. Alternando entre a janela do jogo e a janela da aplicação, que será um navegador *web*;
2. Possuir mais de um monitor e deixar a aplicação aberta em um e o jogo em outro;
3. Utilizar um *Smartphone*, *tablet* ou qualquer outro dispositivo para visualizar a aplicação;

## 1.5. Materiais e Métodos

O projeto foi desenvolvido com a linguagem C# utilizando o *framework* ASP.NET Core na versão 2.0 em conjunto com o Bootstrap e o banco de dados MongoDB.

De acordo com o Time do Bootstrap (2019), o *framework* Bootstrap é um conjunto de ferramentas de código aberto para auxiliar na construção de uma página *web mobile-first* com sistema completos de *grid's* responsivas.

De acordo com Microsoft (2015)

“C# é uma linguagem elegante, orientada a objeto e fortemente tipada, que permite que os desenvolvedores criem uma variedade de aplicativos robustos e seguros executados no .NET Framework. Você pode usar C# para criar aplicativos de cliente do Windows, serviços Web XML, componentes distribuídos, aplicativos cliente-servidor, aplicativos de banco de dados e muito, muito mais.”

A IDE de desenvolvimento utilizada é o Visual Studio Community 2019, que de acordo com Microsoft (2019) é uma IDE grátis e completa para a criação de aplicativos *web*, *mobile* e *desktop*.

A escolha da linguagem C#, do *framework* ASP.NET Core, do Bootstrap e da IDE deveu-se ao fato da experiência do autor na área utilizando estas ferramentas. Uma vez que existem escolhas parecidas optou-se pelas mais conhecidas.

O *framework OpenSource*<sup>3</sup> RiotSharp foi escolhido para tratar as requisições com a *API* da Riot Games, ele trata da parte de limite de requisições por minuto, autorização e respostas da *API*. Algumas contribuições de código foram necessárias para que ela se adequasse bem ao projeto.

O banco de dados utilizado é o MongoDB, um banco de dados orientado a documentos flexível e facilmente escalável. O MongoDB foi escolhido para o projeto devido a flexibilidade de sua estrutura, podendo armazenar documentos de diferentes estruturas em uma mesma coleção (MONGODB INC., 2019). A flexibilidade no armazenamento de dados é interessante pois, conforme o jogo é atualizado, a estrutura dos dados retornados pela *API* é alterada. Como a estrutura é flexível, não é necessário a alteração dos dados existentes.

O Robo3T foi utilizado para montar consultas para serem utilizadas no MongoDB devido a sua simplicidade de uso. De acordo com 3T Software Labs (2017) Robot3T é uma interface gráfica leve grátis para usuários de MongoDB.

A aplicação funcionará baseada nas ferramentas citadas acima, focando nos

---

<sup>3</sup> De acordo com Perens (2007/2008), um *software OpenSource* (código aberto) é um software que pode ser alterado e distribuído por qualquer pessoa de graça.

navegadores Google Chrome, Mozilla Firefox, Safari. Visto que os mesmos são os navegadores mais utilizados atualmente de acordo com StatCounter (2019).

O git, um sistema de controle de versão distribuído que pode ser utilizado para todo tipo de projeto (SOFTWARE FREEDOM CONSERVANCY, 2020), foi utilizado para o controle de versão da aplicação. O repositório git encontra-se hospedado no Github<sup>4</sup> em um repositório público.

Inicialmente no trabalho foram desenvolvidos os requisitos, o projeto de interface e MER, a partir disso o desenvolvimento iniciou-se. Com o tempo o projeto necessitou de alterações e a adoção de uma metodologia foi necessária para manter a organização. A metodologia Kanban foi utilizada, por meio da ferramenta Trello<sup>5</sup>, para dividir o projeto em pequenas tarefas e melhorar a organização. Segundo Anderson (2010), o Kanban permite um controle visual das tarefas, permitindo que seus usuários tenham noção do que já foi feito, do que será feito e do que deve ser priorizado. Geralmente o Kanban é representado por um quadro branco com vários papéis colados. De acordo com Wazlawick (2019), cada papel representa uma tarefa e cada divisão do quadro representa a etapa que ela está. Conforme as tarefas são concluídas elas são movidas pelo quadro de acordo com seu estado. O Trello é basicamente uma representação deste quadro de forma digital, mas que pode ser organizado da maneira desejada. A Figura 2 mostra uma captura de tela que foi realizada do quadro durante o desenvolvimento da aplicação.



**Figura 2. Quadro Kanban utilizado no Trello.**

O projeto consistiu em, primeiramente, observar os dados disponíveis na *API*, análise de trabalhos correlatos e consulta a jogadores. Com isso foi possível definir o que era possível de obter da *API* e que tivesse utilidade ao jogador para ser exibido na ferramenta. A partir daí foram estabelecidos os requisitos com base no que foi definido.

A segunda etapa consistiu na criação dos diagramas de caso de uso e atividade para modelar o projeto da ferramenta.

Com os diagramas estabelecidos, na terceira etapa foram criados protótipos das

<sup>4</sup> Disponível em: <https://github.com/spyker0/fajlol>

<sup>5</sup> Disponível em: <https://trello.com/>

telas utilizando um software de edição de imagem chamado Gimp<sup>6</sup>. Foram desenhadas as telas descritas na Seção 3.4. (“Projeto da Interface”). Logo após a criação dos protótipos, as telas foram desenvolvidas e melhoradas em HTML (*Hypertext Markup Language*) e CSS (*Cascading Style Sheets*), ou seja, a página da *web* propriamente dita.

Após a prototipagem, decorreu a modelagem do banco de dados utilizando a ferramenta lucidchart<sup>7</sup> para tal. Com a modelagem do banco pronta, as tecnologias foram escolhidas e o projeto começou a ser desenvolvido.

## 2. Trabalhos Correlatos

Existem várias soluções que abordam o mesmo problema, dentre elas destacando-se o Porofessor.gg<sup>8</sup> e o U.gg<sup>9</sup>. Ambas funcionam a partir de uma página da internet e tem foco diferente um do outro.

De acordo com o site da própria ferramenta, Porofessor.gg (2020) tem foco na partida atual mostrando informações em tempo real da partida. Ele exibe os jogadores, seu posicionamento nas filas ranqueadas e características de cada um. Dentre as características que referem-se às ações dentro da partida tem-se:

- Principal campeão jogado.
- Como o jogador age durante a partida.
- Se ele está em uma sequência de vitórias/derrotas.
- Com qual campeão ele costuma jogar.
- Proficiência com o campeão escolhido.

A Figura 3 demonstra uma partida analisada pela ferramenta. Na parte superior estão os aliados e na parte inferior os inimigos.

---

<sup>6</sup> Disponível em: <https://www.gimp.org/>

<sup>7</sup> Disponível em: <https://www.lucidchart.com/>

<sup>8</sup> Disponível em: <https://porofessor.gg/>

<sup>9</sup> Disponível em: <https://u.gg/>

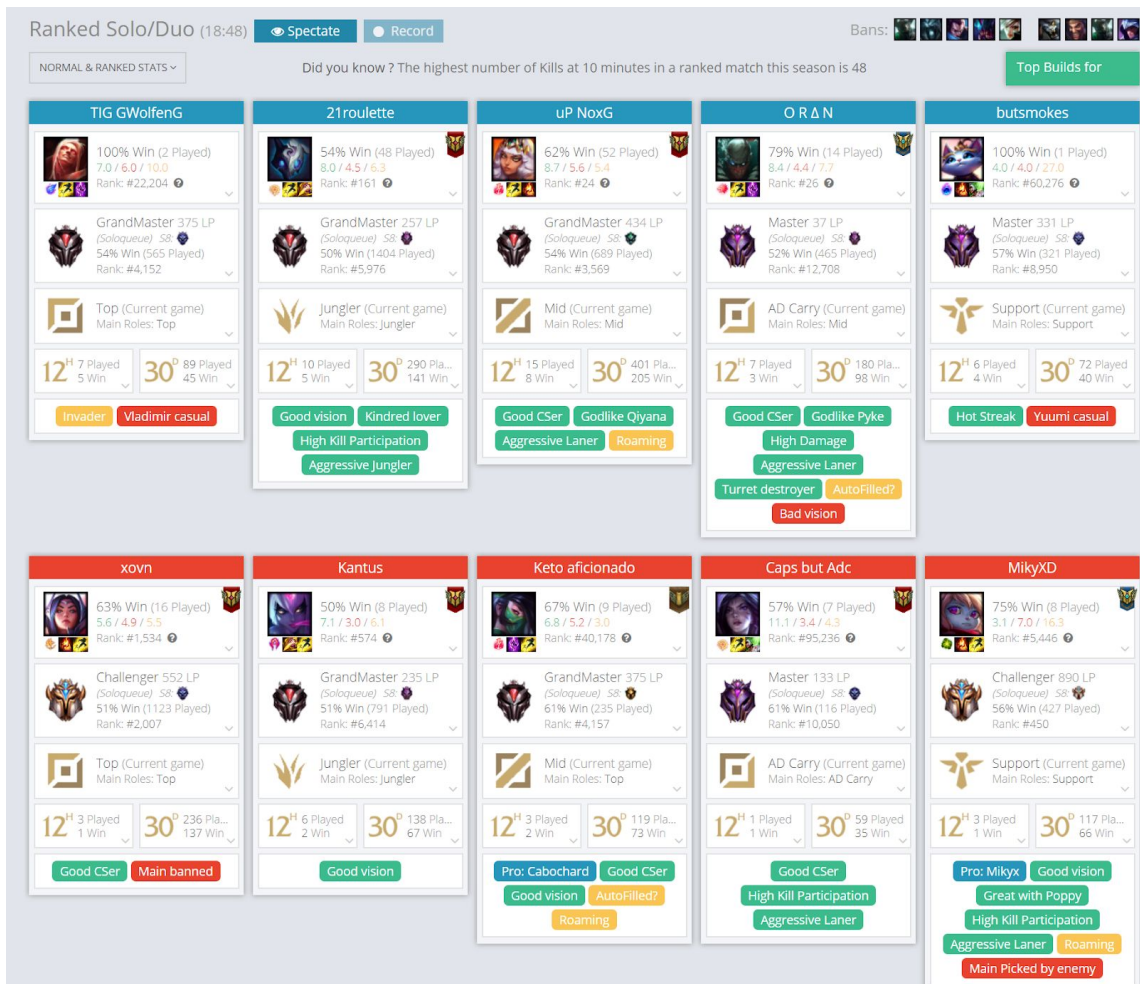


Figura 3. Captura de tela do site Porofessor.gg.

Já o foco do U.gg (Figura 4) é nos campeões (ele até possui informações sobre jogadores, mas essa funcionalidade não está disponível no Brasil no momento). De acordo com os criadores do site (OUTPLAYED INC, 2019), ele reúne várias partidas do jogo e calcula a taxa de vitória de cada campeão, quais itens ele deve fazer e também qual a ordem das habilidades a serem evoluídas para que se tenha chance de melhorar o desempenho durante a partida.



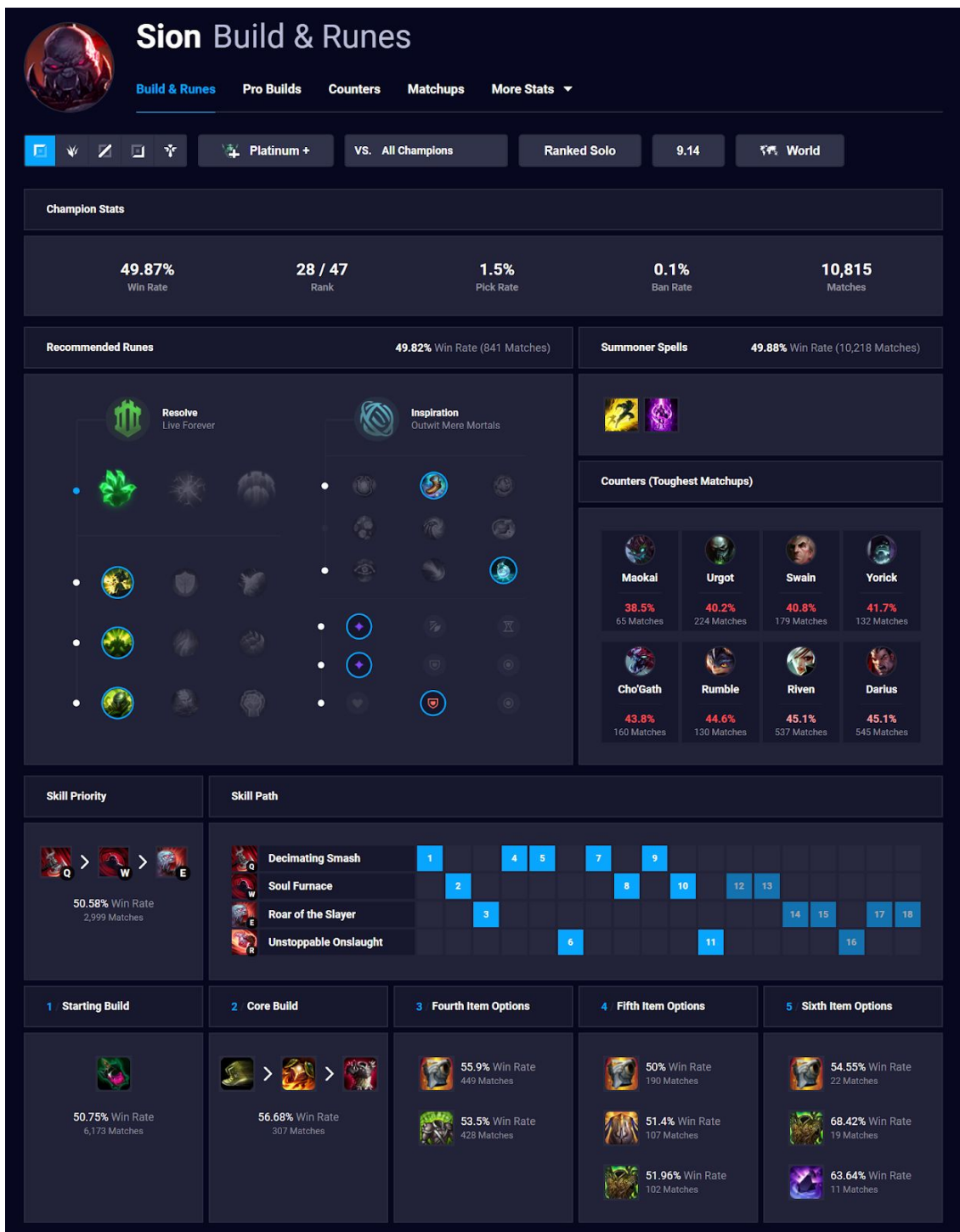


Figura 4. Captura de tela do site U.gg.

O diferencial da ferramenta proposta neste trabalho é a forma com que a informação é apresentada ao usuário. O que a maioria das ferramentas desse tipo faz é exibir dados técnicos ao jogador, sendo que jogadores leigos não são capazes de tirar proveito destes.

A Tabela 1 apresenta um comparativo entre as funcionalidades oferecidas pelo sistema desenvolvido e os trabalhos correlatos. Ao observar-se a tabela nota-se que a



ferramenta proposta apresenta a junção de algumas funcionalidades dos trabalhos correlatos com algumas inéditas.

A chance de vencer a partida permite o jogador ter uma noção do nível de dificuldade que ele irá enfrentar.

As ações a serem tomadas pelo jogador, itens sugeridos e campeões com que o jogador melhor desempenha fornecem informações do que deve ser feito sem que o jogador precise chegar a essas conclusões sozinho.

O principal campeão, características e *lane* principal dos jogadores demonstram quem pode estar fora da sua zona de conforto (sem o principal campeão por algum motivo) e fornecem informações sobre eles.

A taxa de vitória, posicionamento em filas ranqueadas, histórico de partidas e periculosidade/confiabilidade<sup>10</sup> dos jogadores demonstram se eles estão indo bem nas partidas ultimamente e dá uma noção de quem é confiável ou perigoso.

**Tabela 1 — Comparativo de funcionalidades entre a ferramenta desenvolvida e trabalhos correlatos.**

Funcionalidade	Fajlol <sup>11</sup>	Porofessor.gg	U.gg
Chance de vencer a partida	✓		
Ações a serem tomadas pelo jogador (nesse caso, o usuário da ferramenta) durante a partida	✓		
Principal campeão do jogador	✓	✓	
Taxa de vitória do jogador	✓	✓	
Características dos jogadores durante a partida	✓	✓	
Posicionamento dos jogadores nas filas ranqueadas	✓	✓	
Itens sugeridos para a partida	✓		✓
<i>Lane</i> principal do jogador	✓	✓	
Periculosidade/Confiabilidade do jogador	✓		
Histórico de partidas	✓		
Disponível em português	✓	✓	
Campeões que o jogador melhor desempenha	✓		

As funcionalidades do Fajlol, na maior parte, são focadas no que deve ser feito

<sup>10</sup> Respectivamente o quanto deve-se temer ou confiar em um jogador, será explicado mais à frente nas Seções 4.2. e 4.3.

<sup>11</sup> Abreviação para o nome da ferramenta proposta neste trabalho, Ferramenta de auxílio a jogadores de League of Legends.

para ganhar a partida.

### **3. Projeto**

Nesta Seção descreve-se o projeto do Fajlol. A Seção 3.1 aborda os requisitos. A Seção 3.2 aborda diagramas UML. A Seção 3.3 aborda a modelagem de dado. A Seção 3.4 descreve o projeto de interface e a Seção 3.5. aborda os algoritmos desenvolvidos para o Fajlol.

#### **3.1. Requisitos**

Nesta Seção estão expostos os requisitos funcionais e não funcionais do sistema. De acordo com Turine e Masiero (1996),

“A Funcionalidade define os requisitos funcionais que o software ou componentes do software devem executar. A funcionalidade diz respeito à finalidade a que se propõe o produto de software e é, portanto, a principal característica de qualidade para qualquer tipo de software. (...) Os requisitos não funcionais, também denominados de requisitos de qualidade, incluem tanto limitações no produto (desempenho, confiabilidade e segurança) como limitações no processo de desenvolvimento (custos, métodos a serem adotados no desenvolvimento e componentes a serem reutilizados). As características de qualidade de software são: Usabilidade, Confiabilidade, Eficiência, Manutenibilidade e Portabilidade”

##### **3.1.1. Requisitos Funcionais**

- Manter as estatísticas do jogador armazenadas para economizar processamento.
- Desenvolver tela de partida atual.
- Na tela de partida atual:
  - Buscar dados de todos os jogadores da partida atual quando esta for selecionada.
  - Sugerir itens que tiverem maior taxa de vitória com o campeão para o jogador.
  - Caso algum oponente possuir o feitiço com o id 14 (incendiar), avisar para o jogador tomar cuidado com aquele oponente até os 15 minutos de partida.
  - Caso o time inimigo costume matar o primeiro oponente com maior frequência do que o time aliado, avisar o jogador para ter cuidado com confrontos no início da partida.
  - Caso o time inimigo costume matar o Barão (Objetivo do jogo) mais cedo no jogo do que o time aliado, avisar o jogador para manter visão do Barão a partir do 20 minutos de jogo.
  - Exibir nível de periculosidade de cada jogador baseado na taxa de vitória com o campeão, de forma que:
    - Normal: Taxa de vitória abaixo de 50%.
    - Potencialmente perigoso: Taxa de vitória entre 50% e 60%.
    - Perigoso: Taxa de vitória entre 60% e 70%.
    - Muito perigoso: Taxa de vitória acima de 70%.

- Exibir chance de vitória do time baseado em 4.1.
- Exibir jogadores confiáveis do seu time baseado na taxa de vitória, sendo classificados em não confiável (50% ou menos), pouco confiável (entre 50% e 60%) e confiável (Maior que 60%).
- Desenvolvimento de tela inicial.
- Na tela inicial, deve ser possível inserir seu nome e região, e terá a opção de ver o histórico de partidas do jogador ou a partida atual.
- Desenvolvimento da tela de histórico de partidas onde serão exibidas as partidas recentes do jogador.
- A tela de histórico deve conter:
  - Taxa de vitória.
  - Campeões mais utilizados.
  - *Lanes* jogadas e seu percentual.
  - Partidas jogadas.
- Cada partida jogada terá os seguintes dados:
  - Se foi vitória ou derrota
  - Campeão utilizado
  - KDA (*Kill Death Assistant* ou Abate Morte Assistência em português, refere-se ao número de jogadores abatidos, número de mortes e auxílio em abates respectivamente).
  - Posição do jogador
  - Nível no qual terminou a partida
  - Ouro com o qual terminou a partida
  - Itens do jogador
  - Sentinela<sup>12</sup> final do jogador
- Cada partida terá a opção de ter seus detalhes exibidos, sendo eles todos os dados acima para cada jogador, com a adição de:
  - Feitiços de cada jogador
  - *Elo* dos jogadores
  - Farm por minuto dos jogadores
  - Ouro por minuto
  - Número de criaturas chaves mortas por equipe, sendo elas barões e dragões
  - Número de torres destruídas por equipe
- Buscar e armazenar as seguintes informações e todos os seus dados vindos da *API* conforme os jogadores forem requisitando:
  - Dados dos jogadores.
  - Dados das partidas anteriores dos jogadores.
- Calcular os melhores itens para cada campeão (explicado na Seção 3.5.4.).

### 3.1.2. Requisitos Não Funcionais

- Na tela inicial é obrigatório informar o nome de usuário e região do jogador

---

<sup>12</sup> Um tipo de item do jogo que geralmente pode ser colocado no mapa, concedendo visão daquela região sem que o jogador esteja no local.

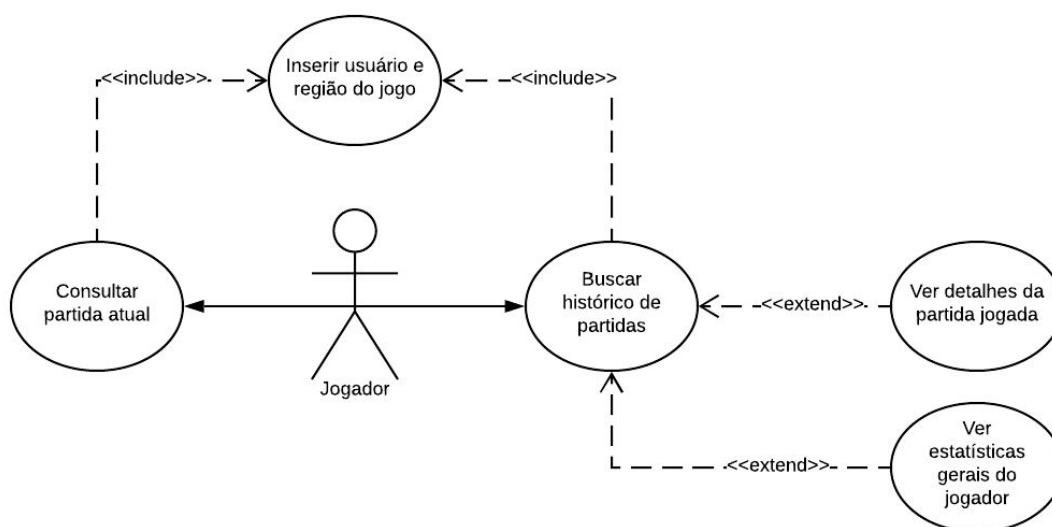
- Servidor Windows ou Linux, com no mínimo 2GB de memória RAM, um processador de 2 núcleos e 100GB de armazenamento.
- Conexão com internet.
- Funcionar em navegadores *web*.

## 3.2. Diagramas UML

A seguir estão os diagramas de caso de uso e de atividade escolhidos para o projeto. O diagrama de caso de uso está presente na Seção 3.2.1. e foi escolhido para ter-se uma noção do que o sistema iria contemplar. Já o diagrama de atividade presente na Seção 3.2.2. foi escolhido para facilitar a visão de fluxo do sistema.

### 3.2.1. Caso de Uso

A Figura 5 representa os casos de uso do sistema. Eles giram em torno do jogador (único ator do sistema) e de seu desempenho, sempre sendo necessário que ele informe usuário e região para que seus dados sejam analisados, ajudando assim em sua partida atual ou exibindo as partidas já jogadas.



**Figura 5. Casos de uso desenvolvidos no projeto.**

### 3.2.2. Diagramas de atividade

A Figura 6 demonstra o fluxo de uso do sistema por meio de um diagrama de atividades, que consiste no jogador inserir seus dados e decidir o que deseja visualizar.

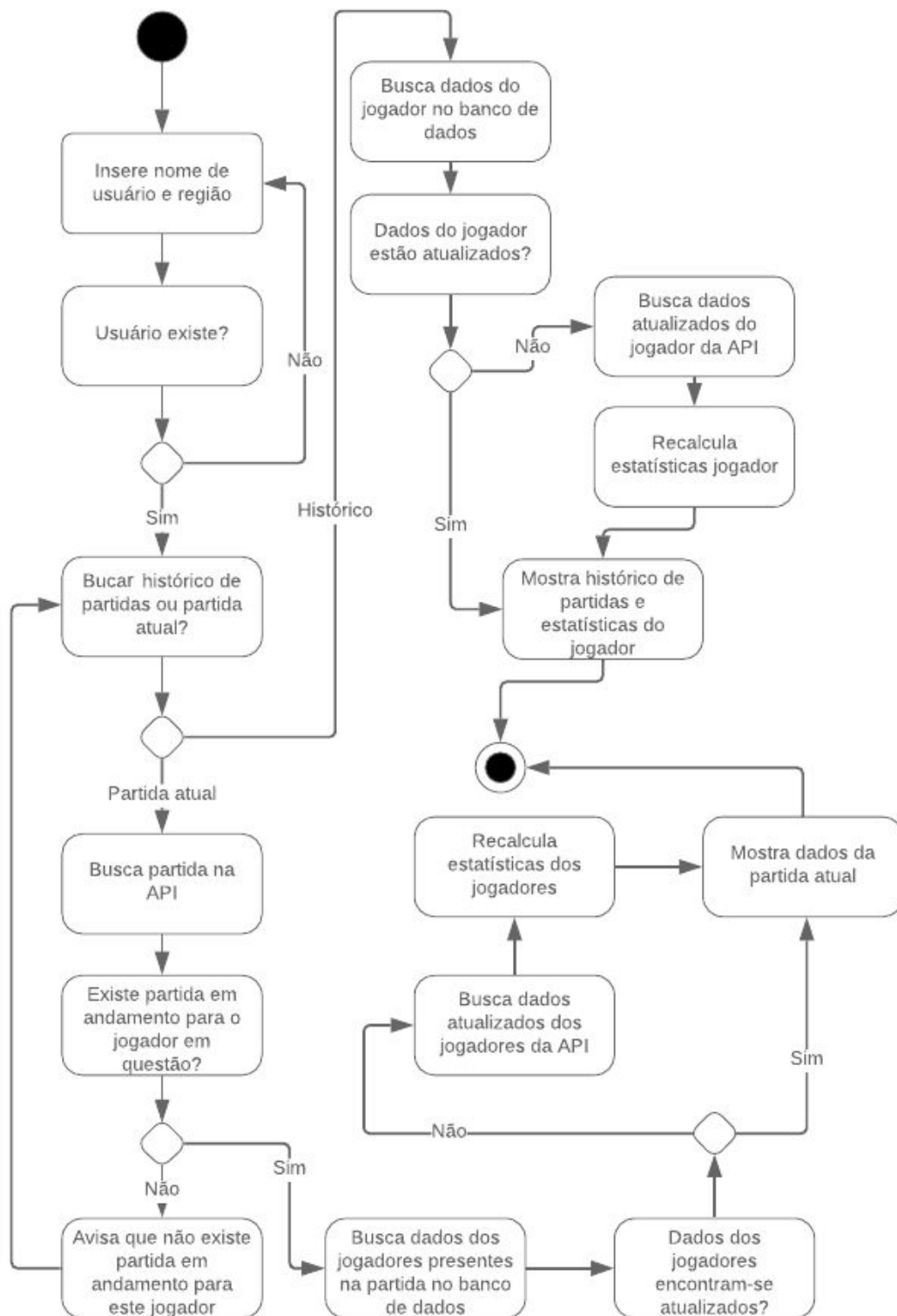


Figura 6. Diagrama de atividade da utilização do sistema.

### 3.3. Modelagem de Dados

Como o MongoDB foi utilizado, não existe relacionamento entre entidades, os dados relacionados são armazenados dentro da mesma entidade.

De acordo com MongoDB, Inc. (2019) o MongoDB armazena documentos flexíveis semelhantes ao JSON (*JavaScript Object Notation* – Notação de Objetos JavaScript). Um JSON consiste em uma coleção de pares nome/valor, estes valores por sua vez podem ser outros objetos JSON (JSON.ORG, 1999). As tabelas, de forma semelhante ao JSON, possuem pares de nome/valor. No caso de EstatisticasJogador por exemplo, em um modelo relacional teríamos a tabela de EstatisticasJogador, uma tabela de *Lanes* e uma tabela de CampeosXTaxaVitoria. As tabelas *Lanes* e CampeosXTaxaVitoria teriam a chave de EstatisticasJogador. No caso de um banco de dados orientado a documentos, o MongoDB neste caso, existe somente a entidade EstatisticasJogador, que possui dentro dela duas propriedades, *Lanes* e CampeosXTaxaVitoria, estes por vez, são coleções de objetos contidas dentro da própria entidade EstatisticasJogador. Como quando obtem-se a entidade EstatisticasJogador as entidades *Lanes* CampeosXTaxaVitoria já estão contidas nela, não precisamos de um relacionamento para encontrar as duas.

O sistema consiste em basicamente quatro entidades principais, sendo elas Partidas (Figura 7.1 e 7.2), EstatisticasJogador (Figura 8), MelhoresItensCampeao (Figura 9) e PartidaAtual (Figura 10). Elas possuem várias entidades filhas.

A entidade Partidas foi baseada nos dados de partidas que são fornecidos pela API da Riot Games e as demais foram baseadas nas especificações do sistema.

As entidades que fazem parte do grupo de Partidas são:

Partidas – Entidade base, possui dados de identificação da partida e entidades filhas.

ParticipantIdentities – Possui a identificação dos jogadores na partida e a entidade filha Player.

Player – Possui identificadores universais dos jogadores, de modo que é possível identificar eles em qualquer partida.

Teams – Contém dados referentes a cada time.

Bans – Contém os campeões que foram banidos<sup>13</sup> por cada time.

Participants – Possui informações referentes às escolhas de cada jogador antes de iniciar a partida.

Timeline – Possui informações referentes ao que cada jogador fez no decorrer da partida, com várias informações separadas por grupos de 10 minutos.

Stats – Possui informações referentes ao que cada jogador fez durante a partida, como a primeira vez que algo ocorreu ou o total de dano infligido ao adversário.

As entidades que fazem parte do grupo de EstatisticasJogador são:

EstatisticasJogador – Entidade base, possui dados de identificação do jogador

---

<sup>13</sup> Termo utilizado para definir ação de um jogador que consiste em proibir a utilização de um campeão na partida

algumas estatísticas calculadas.

CampeoesXTaxaVitoria – Contém dados referentes a campeões e a taxa de vitória do jogador com o mesmo.

Lane – Contém dados referentes as *lanes* que o jogador participa.

As entidades que fazem parte do grupo de MelhoresItensCampeao são:

MelhoresItensCampeao – Entidade base, possui o id do campeão e os melhores itens e sentinela para tal.

Item – Dados referentes ao item ou sentinela.

As entidades que fazem parte do grupo de PartidaAtual são:

PartidaAtual – Entidade base, possui dados de identificação da partida e algumas estatísticas para o jogador principal.

Jogador – Possui dados de cada jogador da partida atual.

Dicas – Contém as dicas que foram fornecidas ao jogador.

Item – Itens sugeridos ao jogador.

Participant – Possui informações referentes às escolhas de cada jogador antes de iniciar a partida.

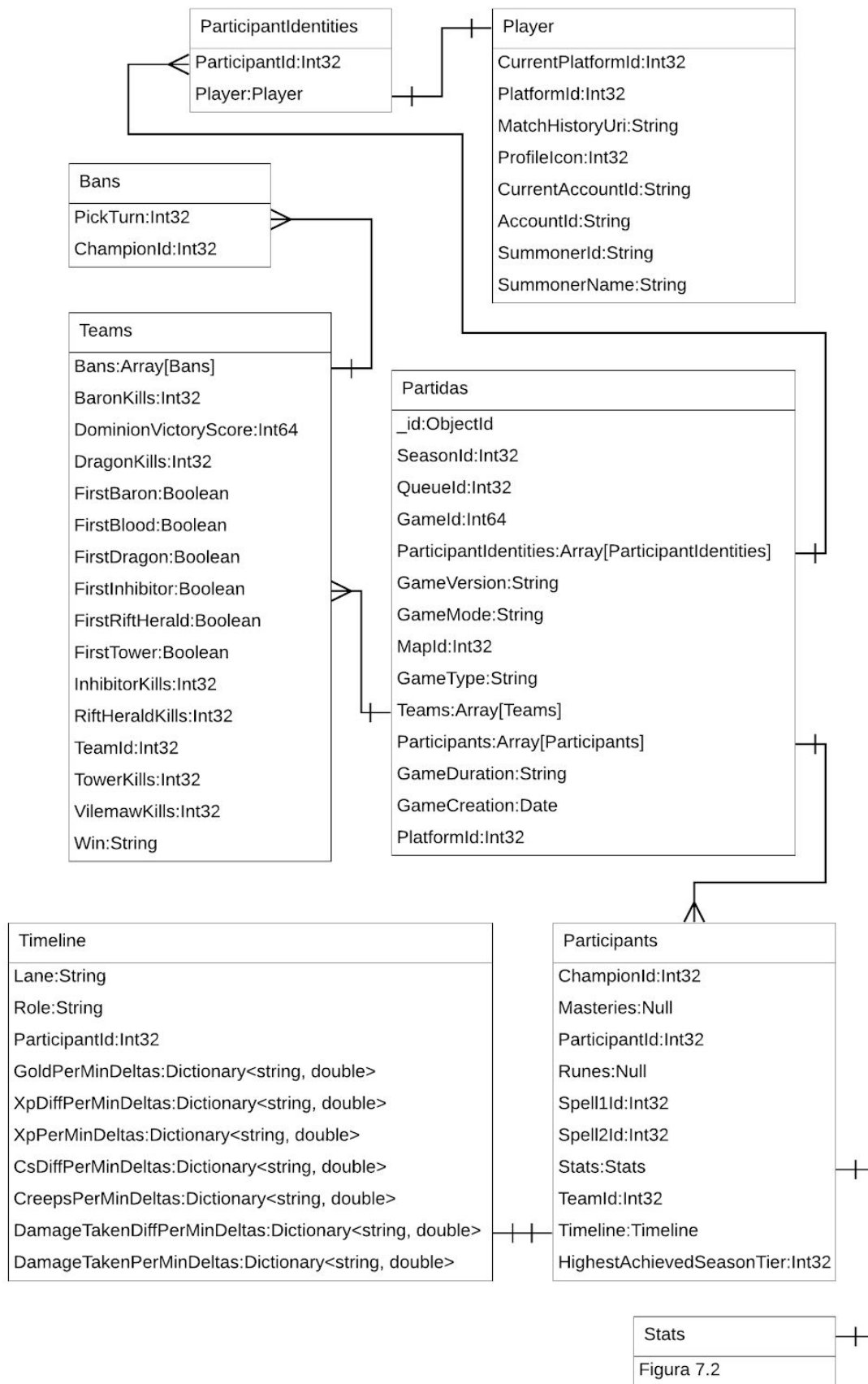
Perks – Contém dados referentes às runas<sup>14</sup> escolhidas para a partida.

Nas Figuras 7.1, 7.2 e 8 estão representadas as entidades que são utilizadas pelo sistema. São apresentados os nomes dos campos, tipo e cardinalidade com entidades filhas.

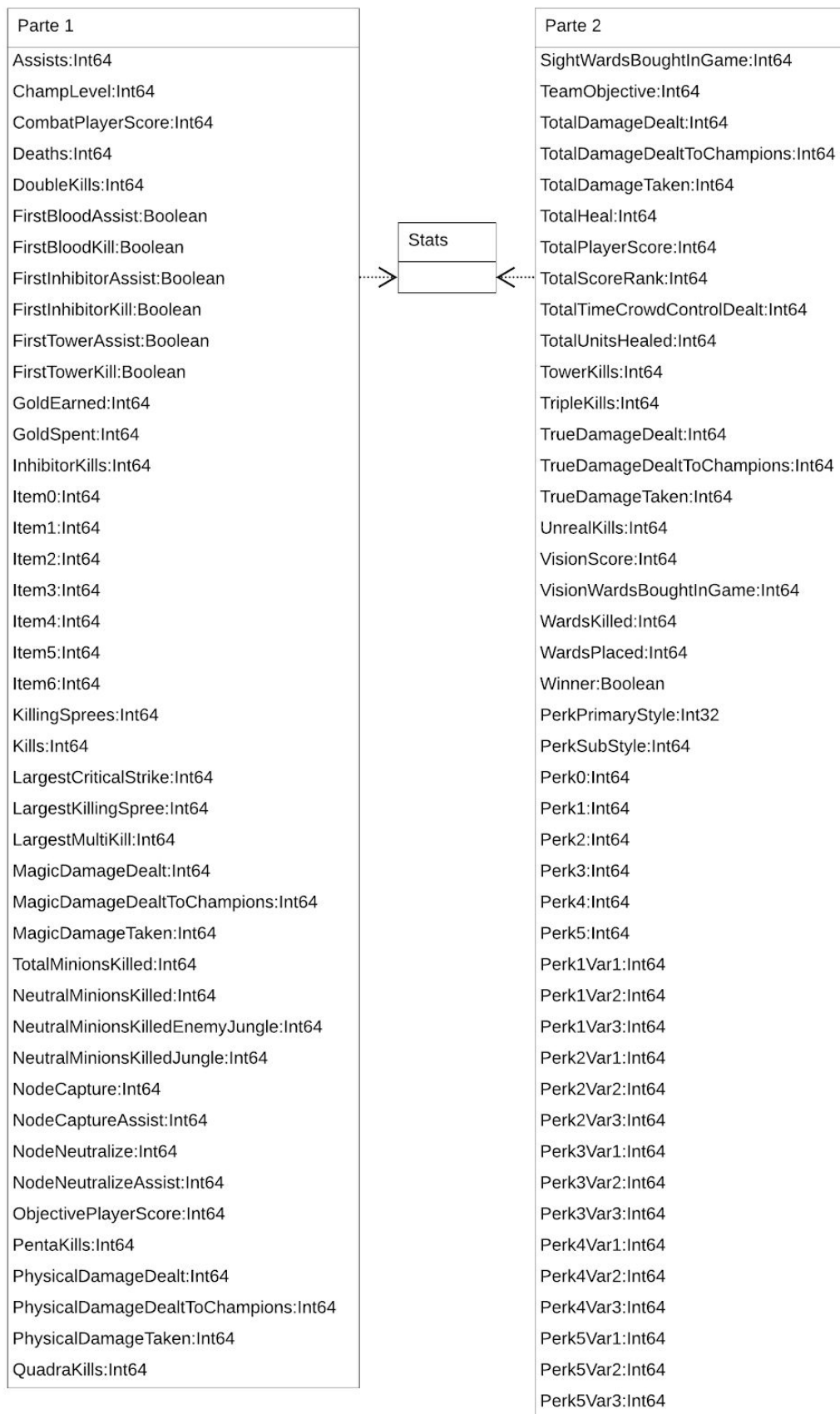
---

<sup>14</sup> Runas permitem ao jogador adicionar ou melhorar habilidades, efeitos passivos e bônus a seus campeões. Elas podem ser utilizadas para customizar o campeão antes da partida (FLORES, 2018).

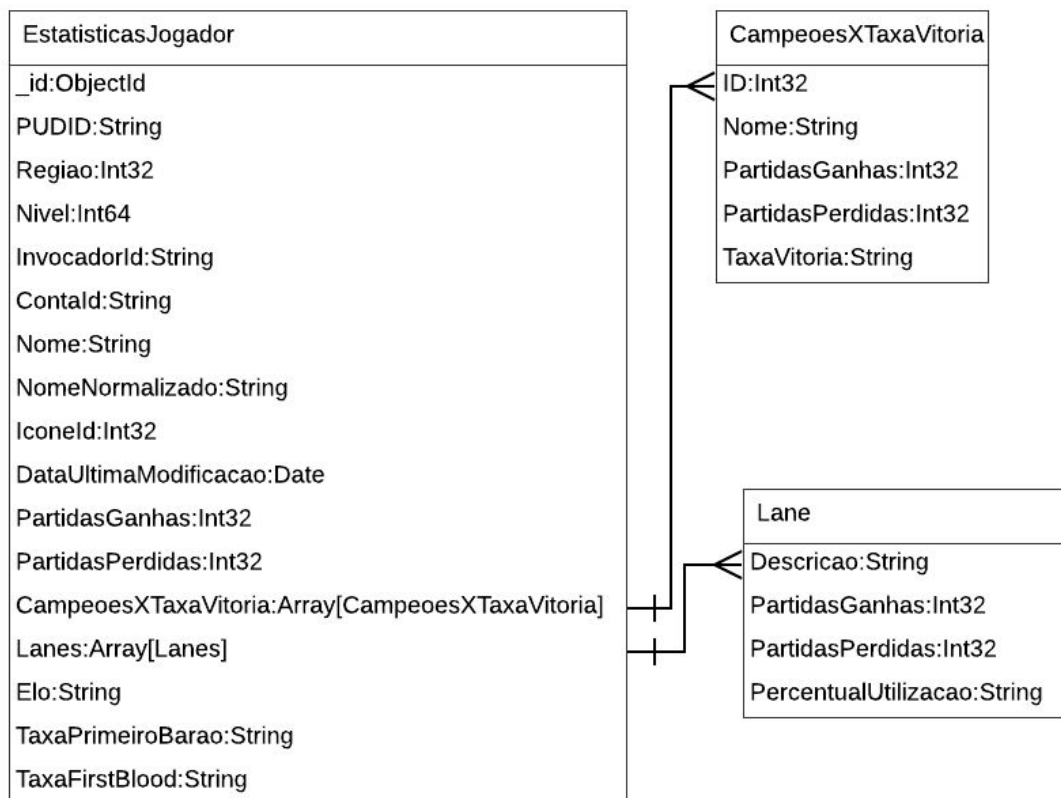




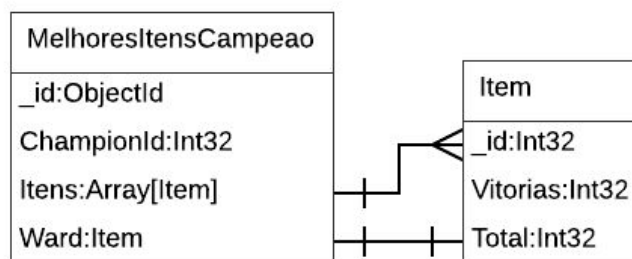
**Figura 7.1. Entidade Partidas.**



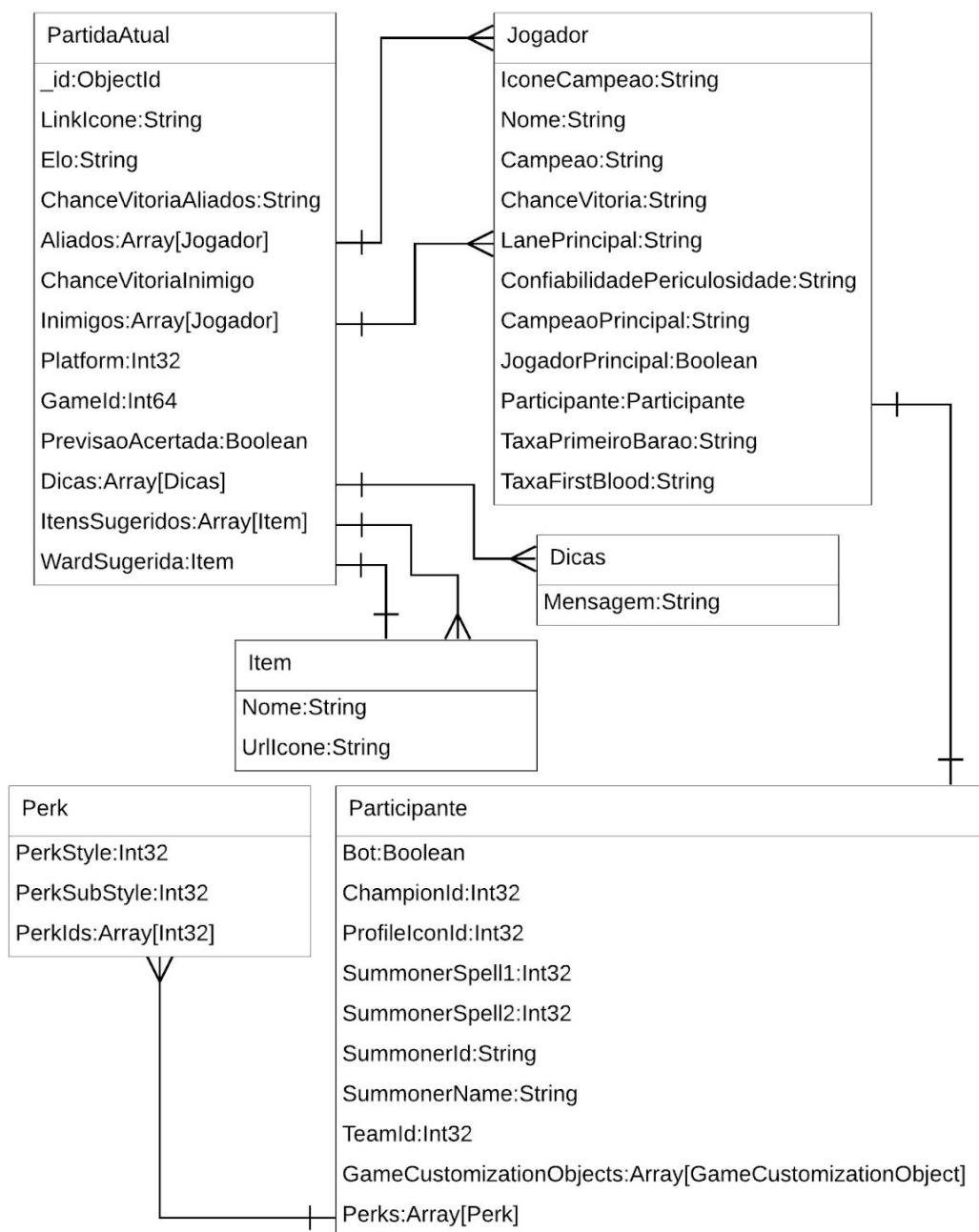
**Figura 7.2. Entidade Partidas - Stats.**



**Figura 8. Entidade EstatisticasJogador.**



**Figura 9. Entidade MelhoresItensCampeao.**



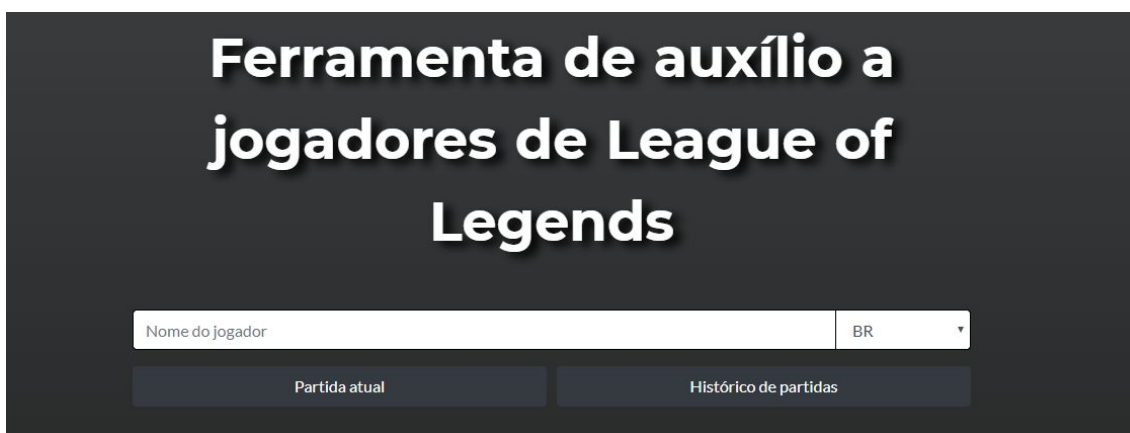
**Figura 10. Entidade PartidaAtual.**

### 3.4. Projeto da Interface

O design do sistema foi pensado para que o usuário tenha todas as informações mais importantes disponíveis na tela ao mesmo tempo. O Bootstrap auxilia o sistema a se manter responsivo e ter um bom funcionamento em dispositivos móveis.

A Figura 11 demonstra a tela inicial, na qual o usuário deve primeiro informar o

nome do seu jogador e a região em que joga. Estes dados são necessários para que o jogador possa visualizar suas estatísticas. Logo após ele tem a opção de visualizar a partida atual ou o histórico de partidas.



**Figura 11. Página inicial.**

A Figura 12 mostra a página de partida atual, que tem a finalidade de auxiliar o jogador com a partida em andamento.

No cabeçalho da tela esquerda é exibido o nome do projeto, clicar nele leva o usuário a tela inicial. A direita são exibidos o nome e região do jogador.

No topo da página no lado esquerdo são exibidos os itens sugeridos com base nos requisitos e a direita algumas dicas baseadas também nos requisitos.

Por fim, tem-se dois painéis com aliados e inimigos. Os dois possuem informações bem parecidas. Ambos possuem a chance de vencer do lado do título do painel. Os aliados possuem nome, campeão, divisão, confiabilidade, *lane* principal e campeão principal. O jogador principal não tem sua confiabilidade exibida, visto que este é um fator que implica sobre outros jogadores somente. Os inimigos possuem o nome, campeão, divisão, periculosidade, *lane* principal e campeão principal (a diferença fica por conta da troca de confiabilidade por periculosidade).

TCC

Partida atual de spyker0 - BR1

**Itens sugeridos:**

**Dicas:**

- Mantenha distância do jogador the marimbondo até os 15 minutos, pois ele possui o feitiço incendiar, que pode ser mortal caso você esteja com pouca vida.
- Cuidado com confrontos no início da partida, o time inimigo costuma matar o primeiro oponente antes do seu!

**Aliados - Chance de vencer - 25.40%**

	<b>Krastyl Fanboy</b> Campeão: Vladimir Divisão: CHALLENGER I	Você! Lane principal: Mid Campeão principal: Syndra
	<b>tochinoshinn</b> Campeão: Thresh Divisão: CHALLENGER I	Não confiável Lane principal: Support Campeão principal: Thresh
	<b>griff24</b> Campeão: Ezreal Divisão: GRANDMASTER I	Não confiável Lane principal: Adc Campeão principal: Aphelios
	<b>The Elysium</b> Campeão: Olaf Divisão: GRANDMASTER I	Não confiável Lane principal: Top Campeão principal: Aatrox
	<b>dogla la dogla</b> Campeão: Rumble Divisão: GRANDMASTER I	Não confiável Lane principal: Top Campeão principal: Rumble

**Inimigos - Chance de vencer - 74.60%**

	<b>Victor carry üs</b> Campeão: Fiora Divisão: GRANDMASTER I	Muito perigoso Lane principal: Top Campeão principal: Fiora
	<b>MASTER YI DOENTE</b> Campeão: Jarvan IV Divisão: CHALLENGER I	Potencialmente perigoso Lane principal: JUNGLE Campeão principal: Master Yi
	<b>Cabeça de navio</b> Campeão: Diana Divisão: GRANDMASTER I	Potencialmente perigoso Lane principal: Mid Campeão principal: Diana
	<b>the marimbondo</b> Campeão: Braum Divisão: CHALLENGER I	Muito perigoso Lane principal: Support Campeão principal: Morgana
	<b>Clair de Lune</b> Campeão: Senna Divisão: CHALLENGER I	Muito perigoso Lane principal: Adc Campeão principal: Aphelios

**Figura 12. Página da partida atual.**

A Figura 13 demonstra a página de histórico de partidas, bem como na tela de partida atual, no cabeçalho da tela esquerda é exibido o nome do projeto, clicar nele leva o usuário a tela inicial.

No topo da página tem-se o ícone do jogador, seguido por seu nome, região e taxa de vitória. No centro temos as posições do jogador junto ao percentual de atuação em cada uma. A direita temos os campeões mais utilizados pelo jogador, contendo o nome do campeão, número de partidas e taxa de vitória para cada um.

No corpo da página tem-se uma listagem de partidas. Cada partida possui o ícone do campeão que o jogador utilizou na partida, se ele venceu a partida, o campeão que utilizou, a data da partida, seu KDA, posição jogada, nível no fim da partida, ouro acumulado durante a partida, itens e sentinela com qual o jogador finalizou a partida. Também existe um botão ao final de cada linha para exibir mais detalhes da partida.

Nos detalhes da partida, tem-se os mesmos dados para cada jogador, com adição de algumas informações descritas nos requisitos.

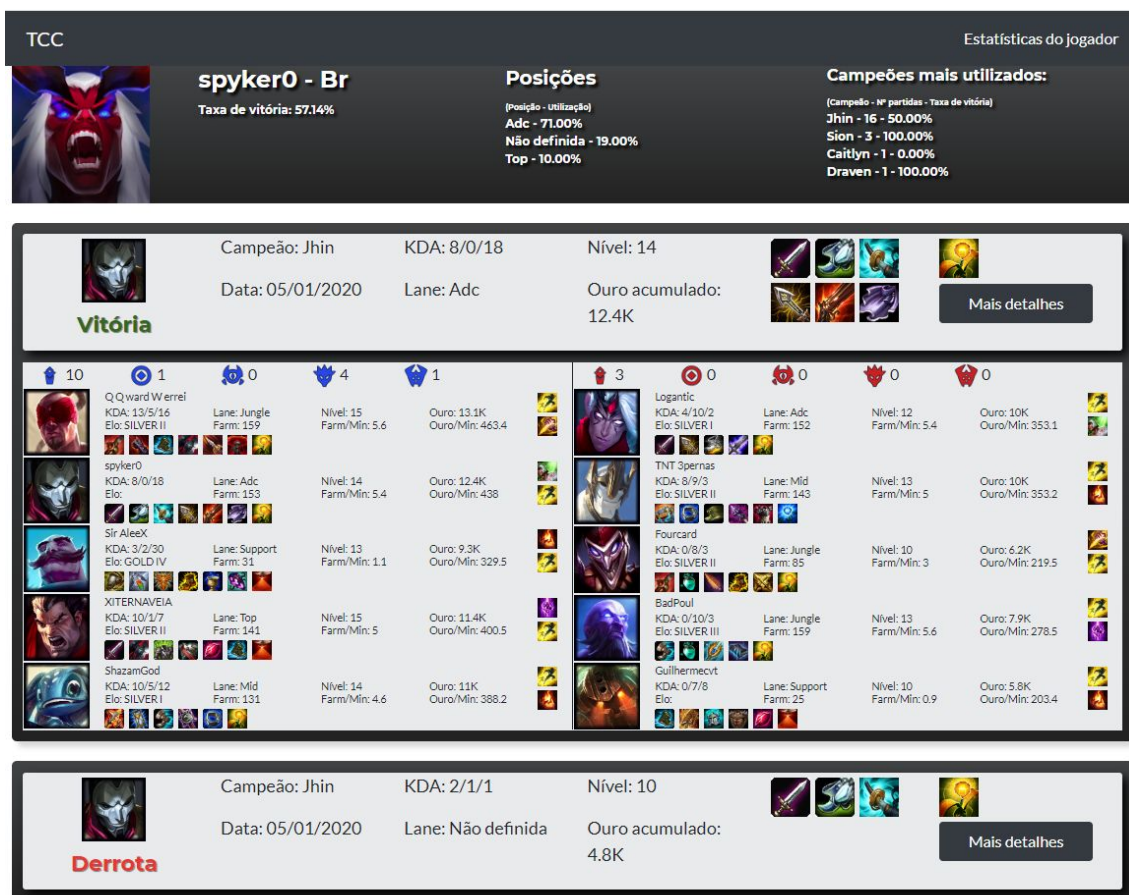


Figura 13. Página de histórico de partidas.

### 3.5. Algoritmos desenvolvidos

A Riot Games fornece dados sobre jogadores, partidas em andamento, campeões, entre outros sobre o jogo. Esses dados são fornecidos de maneira gratuita por meio de uma API aberta, com limite de usos por tempo/requisições. Todas essas informações são necessárias para poder-se gerar informação útil sobre o jogo e a partida em andamento.

Quando a partida estiver iniciando, pode-se buscar os dados dela, onde tem-se os nomes dos jogadores dos dois lados. Com isto em mãos, é possível ter acesso ao histórico de partidas, campeões e posições escolhidas por cada jogador. Combinando os dados obtidos, pode-se gerar informações que podem ser úteis para o jogador durante a partida. Para a geração dessas informações, alguns algoritmos devem ser aplicados, sendo explicados a seguir:

#### 3.5.1. Chance de vencer a partida

No decorrer do trabalho o cálculo de chance de vencer a partida demonstrou-se complexo. Há muitos artigos sobre a chance de vitória em futebol e outros esportes. A maioria deles, como o artigo de Alves et al. (2011), utilizam regressão logística, fazendo simulações de partidas entre os times, contando com derrotas e vitórias de cada um durante o campeonato. O problema desse modelo é que ele considera que os times que se enfrentarão durante o ano serão definidos no início do campeonato. Isso os torna



inviáveis para serem aplicados no League of Legends, pois cada partida conta com 10 jogadores diferentes em cada time.

De acordo com os estudos realizados por Huang, Kim e Leung (2015) e Yin (2018), ambos utilizando *machine learning*, foram utilizadas diversas partidas para treinar a inteligência artificial utilizando diferentes métodos. O que provou-se mais preciso em ambos os casos foi a taxa de vitória do jogador com um campeão específico, ou seja, a proficiência do jogador com um campeão específico é mais importante que qualquer outro detalhe no jogo (como qual o campeão ou quais os campeões do time inimigo). Em um estudo realizado por Kim, Keegan, Park e Oh (2015), onde foram analisadas cerca de 1,9 milhão de partidas, foi provado que o que mais influencia na vitória de uma equipe é a proficiência do jogador em uma posição e campeão.

Em vários artigos em que essa chance de vencer a partida é calculada com maior precisão a inteligência artificial é utilizada.

Como provou-se nos trabalhos citados, o que mais importa é a proficiência do jogador com o campeão, baseado na sua taxa de vitória com ele.

Utilizar machine learning mostrou-se demorado e não prático para a exibição de dados aos jogadores (estes precisam da informação rapidamente). O que foi desenvolvido e será testado é um cálculo considerando a taxa de vitória da pessoa com o campeão

O modelo foi inspirado na entrevista do professor Costa (2016), que consiste em obter uma média entre a taxa de vitória do time A e a taxa de derrota do time B, o que resulta na chance de vitória do time A. Como o League of Legends não é composto por times fixos, foi necessário calcular a taxa de vitória para cada jogador, para assim termos uma taxa de vitória para cada time. No primeiro time foram somadas as taxas de vitória de cada pessoa com o seu campeão, e foi obtida uma média delas (TV) e no segundo time foram somadas as taxas de derrota com seus campeões e feita a média também (TD). Depois disso é obtida uma média de taxa de vitória do time A com a taxa de derrota do time B  $((TV+TD)/2)$ , que resulta na chance do time A vencer a partida. Independente do time de qual foi obtida a taxa de vitória ou derrota o resultado é o mesmo.

### 3.5.2. Confiabilidade

É uma característica atribuída a um aliado do time baseado em sua taxa de vitória com o campeão que está utilizando na partida atual. Ele deve ser usado para o jogador saber o que deve fazer quando cada aliado fazer um *call* (termo comumente utilizado pela comunidade para quando um jogador indica o que o outro deve fazer dentro do jogo).

Segundo Riot Games (2020) o sistema de balanceamento de partidas tenta equilibrar com pessoas que joguem parecido com você, quando o jogador se destaca, ele sobe de *elo*, e quando vai mal cai. Conclui-se então, que para o jogador manter-se em seu nível ele tenha uma taxa de vitória de 50%. Logo, um jogador que possui taxa de vitória de 50% teoricamente equipara-se com você. Assim foi considerada uma taxa de vitória de 50% ou menor como fraca ou equiparada. De acordo com Ucla (2015), treinador e jogador experiente de alto nível, geralmente quando o jogador tem uma taxa

de vitória de 60% ou mais, ele é um *smurf*<sup>15</sup>. A partir disso, foram designados três tipos de confiabilidade:

1. Não confiável (50% ou menos) — deve-se evitar qualquer sugestão oferecida por este jogador.
2. Pouco confiável (entre 50% e 60%) — deve-se analisar a situação antes de seguir a sugestão deste jogador, ele é considerado como um jogador que está desempenhando bem em seu *elo* atual, por estar acima do normal (50%) e abaixo de uma taxa alcançada por um *smurf*.
3. Confiável (Maior que 60%) — esse jogador está acima da média dos outros jogadores da partida e provavelmente é um *smurf*, os seus conselhos devem ser seguidos.

### 3.5.3 Periculosidade

A exemplo da confiabilidade, é uma característica atribuída a um jogador do time inimigo baseado em sua taxa de vitória com o campeão que está utilizando na partida atual. Ele deve ser usado para o jogador saber se deve evitar duelos<sup>16</sup> contra este jogador na partida.

Foram divididas em 4 tipos:

1. Normal (50% ou menos) — pode-se vencer um duelo facilmente contra este jogador.
2. Potencialmente perigoso (entre 50% e 60%) — deve-se analisar a situação antes de duelar com esse jogador
3. Perigoso (entre 60% e 70%) — esse jogador está acima da média dos outros jogadores da partida, não se deve duelar com ele. Seguindo a mesma lógica da confiabilidade, esse jogador provavelmente é um *smurf* também.
4. Muito perigoso (maior que 70%) — deve-se evitar estar na mesma parte do mapa que este jogador, ele tem as habilidades acima do resto dos jogadores da partida. Essa categoria foi criada para um jogador que estaria extremamente acima do nível da partida, com a adição de um percentual em cima do considerado *smurf*.

### 3.5.4. Sugestão de itens

Como o jogo conta com diversos campeões e itens, a sugestão de melhores itens teve de ser calculada em uma rotina separada, que leva algum tempo para ser finalizada. Essa rotina deve ser chamada de tempos em tempos (atualmente ela pode ser chamada manualmente por meio de uma requisição *http*).

A análise foi feita em cima dos campeões e os itens que resultam em maior número de vitórias para cada um baseado nas partidas armazenadas na base de dados.

Para entender o cálculo é necessário entender como os itens são armazenados, basicamente o jogador pode ter no máximo 6 itens e 1 sentinela. Nas partidas

---

<sup>15</sup> De acordo com Coleman (2019), *smurf* é um jogador que possui uma conta de alto nível e que cria uma nova conta para enfrentar jogadores de baixo nível. Dessa forma o jogador estará com uma grande vantagem sobre os novatos por ter mais experiência. Muitas vezes o *smurf* sozinho pode fazer com que seu time vença.

<sup>16</sup> Situação em que dois jogadores, um aliado e um inimigo se enfrentam sem os demais jogadores de cada time.

armazenadas, temos os itens e a sentinela que o jogador possuía no final da partida em cada posição. Para cada posição foram calculados os itens com maior número de vitórias para cada campeão e que não constavam na posição anterior (para não sugerir itens repetidos ao jogador). Lembrando que a ordem dos itens pode influenciar, por isso é importante calcular o melhor item para cada posição, de 1 a 6.

O melhor item para cada posição e a melhor sentinela, todos calculados para cada campeão do League of Legends foram armazenados no banco de dados e são sugeridos ao jogador baseado no campeão que ele está utilizando na partida atual.

#### 4. Implementação

A implementação da ferramenta passou basicamente pelos seguintes passos:

1. A criação de uma conta de desenvolvedor na Riot Games para ter-se acesso às *API's*.
2. Um projeto em ASP.NET Core foi criado.
3. As telas foram criadas com conteúdo fictício para o desenvolvimento dos estilos.
4. As entidades do MER foram criadas como classes dentro do projeto.
5. Os índices necessários para o MongoDB foram configurados dentro da aplicação.
6. Após a inserção do primeiro registro fictício na base, o banco todo foi criado automaticamente e tem seus índices gerados conforme configurado.
7. Os dados necessários foram obtidos da *API*.
8. Primeiro a tela inicial foi desenvolvida.
9. Logo após isso, a tela de histórico foi desenvolvida, obtendo-se da *API* as partidas do jogador e tendo os cálculos necessários desenvolvidos para exibir as estatísticas desta tela.
10. Foram desenvolvidos os cálculos necessários para serem exibidos na partida atual, descritos na Seção 3.5. (“Algoritmos desenvolvidos”).
11. Os cálculos desenvolvidos foram aplicados junto de outros cálculos básicos.
12. As informações e cálculos obtidos foram exibidos na tela de partida atual.
13. Os estilos das telas foram ajustados novamente.
14. O Fajlol foi testado com os jogadores.

A aplicação não foi publicada em um link público pois ainda não tem uma chave de produção. Segundo Riot Games Inc (2020) uma das regras de utilização da *API* é não usar a uma chave de desenvolvimento para uma aplicação que esteja disponível publicamente.

O desenvolvimento contou com várias dificuldades, principalmente no cálculo de chance de vencer a partida. Não existia nenhum modelo pronto para ser implementado que possa gerar um resultado rapidamente. A maioria das outras estatísticas também não tinha uma fórmula pronta, sendo necessário recorrer a criadores de outras ferramentas ou ao desenvolvimento de novos cálculos.

A escassez de documentação da *API* de League of Legends também gerou algumas dificuldades no projeto.

Algumas dificuldades foram encontradas também com a utilização do *framework* RiotSharp, que continha algumas implementações com *bugs*, fazendo-se necessárias algumas contribuições com o código da biblioteca.

#### 5. Resultados Obtidos

Nesta Seção descrevem-se os resultados obtidos no trabalho. A Seção 5.1. fala sobre os resultados do cálculo desenvolvido da chance de vencer a partida e a Seção 5.2. fala sobre os resultados obtidos na aplicação do Fajlol com jogadores, com eles jogando e utilizando a ferramenta.

### 5.1. Experimento

Para o teste de cálculo de chance de vencer a partida, foi desenvolvido um método que faz as seguintes ações:

1. Obtém uma partida em destaque ocorrendo no momento.
2. Obtém todos os jogadores da partida.
3. Obtém as últimas quatro partidas jogadas com o campeão que cada jogador está utilizando na partida em andamento.
4. Calcula a chance de vencer a partida para cada jogador do time e também uma média para o time, como descrito na seção 3.5.1. (“Chance de vencer a partida”).
5. Salva estas informações no banco de dados.

O programa realizou previsões de 124 partidas em andamento durante 6 horas. Um total de 4394 partidas foram obtidas da *API* para o cálculo de chance de vencer a partida. É preciso levar em consideração que alguns jogadores estavam presentes em mesmas partidas no passado, por isso, mesmo obtendo 4 partidas de cada jogador em 124 jogos em andamento, o que totaliza 4960 partidas anteriores, foi gravado somente um total de 4394.

Após a previsão, foi obtido o resultado das partidas e comparado com a previsão. Caso a previsão de vitória fosse de mais de 50% e o jogador tivesse ganho a partida, a previsão é considerada um sucesso, do contrário é considerado uma falha. Nenhuma partida teve um empate como previsão (exatamente 50% de chance), de qualquer forma, o empate não existe no League of Legends.

Das 124 partidas com resultado previsto, 67 tiveram seu resultado previsto com êxito e consequentemente 57 não foram previstas com sucesso. Totalizando assim uma taxa de 54,03% de acerto da fórmula desenvolvida. A Tabela 2 compara a precisão da previsão do Fajlol com a de outras ferramentas.

**Tabela 2 — Comparativo de precisão de previsão entre as ferramentas.**

<b>Fajlol</b>	<b>Yin (2018)</b>	<b>Huang, Kim e Leung (2015)</b>
54,03%	60,00%	92,80%

A média de chance de vitória do time aliado das partidas que não foram previstas com sucesso é de 49,4%, o que justifica um pouco os erros, porque, quando um time tem uma chance de 49,4% de vencer, mesmo que a chance de ganhar ou vencer seja praticamente a mesma, a ferramenta considera uma derrota.

A previsão leva cerca de 30 segundos, foi adicionado um tempo extra entre as obtenções de partida para evitar que o limite de requisições fosse atingido.

Conforme a documentação da *API* (RIOT GAMES INC, 2020) o limite de

requisições para desenvolvimento é de 20 requisições a cada 1 segundo e no máximo 100 requisições a cada 2 minutos, isso faz com que em desenvolvimento a predição de resultado de uma partida demore cerca de 30 segundos.

Caso o projeto for aprovado para produção pela Riot Games, o limite de requisições fica consideravelmente maior, com 3,000 requisições a cada 10 segundos e um total de 180,000 a cada 10 minutos, o que permite que seja feito o download de mais partidas, tornando o cálculo mais preciso e mais rápido, praticamente instantâneo.

## 5.2. Avaliação e Resultados

O Fajlol foi testado com três jogadores diferentes em um total de quatro partidas, sendo quatro análises cada um. Os três jogadores encontravam-se há cerca de um mês sem jogar e jogaram todas as partidas juntos. Eles também jogam League of Legends há alguns anos. Cada partida levou cerca de 50 minutos.

Como podemos observar na Tabela 3, em 50% das partidas o Fajlol acertou a previsão. Subindo para 55,6% se considerarmos somente as vezes em que o jogador jogou com o campeão recomendado pela ferramenta. Em 66,7% das partidas em que os jogadores seguiram as indicações da ferramenta (foram seguidas em todas as partidas por todos os três jogadores) e jogaram com um campeão recomendado eles venceram a partida.

O jogador número 1 jogou quatro partidas com o mesmo campeão, mantendo sua taxa de vitória com ele em 50% (que tinha antes de parar de jogar). Na última partida em que todos os três jogadores seguiram as orientações da ferramenta e jogaram com os campeões recomendados, a partida foi vencida e a ferramenta acertou a previsão.

**Tabela 3 — Resultados dos testes do Fajlol.**

Partida	Jogadores com campeão recomendado	Venceu?	Ferramenta acertou?
1	2/3		
2	2/3	✓	✓
3	2/3		
4	3/3	✓	✓

De acordo com os jogadores, a ferramenta apresentou itens satisfatórios. Ainda segundo eles, as dicas são realmente voltadas para iniciantes, que para eles, jogadores mais experientes, não são de extrema valia. E finalmente, a confiabilidade e periculosidade demonstrou-se interessante segundo os jogadores, ajudou a evitar combates indesejados.

## 6. Considerações Finais e Trabalhos Futuros

Com o desenvolvimento e testes da ferramenta, nota-se que o League of Legends é um

jogo complexo e cheio de variáveis. A quantidade de dados disponíveis abre espaço para a geração de muita informação. O desafio fica em obter informação que é realmente útil e que seja clara aos jogadores.

O sistema facilita em alguns pontos, principalmente em manter toda a informação de forma simples ao usuário e afirmando o que ele deve fazer e qual campeão utilizar em suas partidas. A outra parte da evolução depende do usuário, como jogar assiduamente e seguir o que a ferramenta propõe. Nos testes notou-se alguma resistência dos jogadores para seguirem as instruções, principalmente de qual campeão jogar. Isso pode acontecer pois, nem sempre o jogador deseja vencer e se tornar melhor, mas dá mais importância à diversão.

No decorrer da pesquisa notou-se que para algumas estatísticas serem calculadas são necessárias algumas formas de coletas de dados que não estão claras na documentação da *API*, tornando uma pesquisa com os criadores de ferramentas semelhantes necessária numa próxima etapa. Um exemplo disso é o cálculo da taxa de vitória de um campeão no geral e itens mais utilizados por campeão em posição específica. Para isso seria necessário coletar uma grande quantidade de partidas e todos os detalhes delas.

No trabalho foram analisadas cerca de 4960 partidas, o U.gg, um site grande com uma chave de produção já citado no trabalho, analisou cerca de 11 milhões de partidas em entre os dias 18 e 28 de fevereiro para montar as análises. Para obter-se esse número de partidas, seria necessário submeter a aplicação a uma análise para a Riot Games possibilitando a obtenção de uma chave de produção (caso a ferramenta se mostre útil para a produtora).

A ferramenta obteve uma taxa de acerto interessante em partidas em que o jogador seguiu todas as orientações dela (55,6%). Seriam necessários mais testes, principalmente com uma chave de produção e com um número maior de pessoas para solidificar essa taxa.

Na próxima etapa de desenvolvimento deste aplicativo, deseja-se desenvolver de uma versão para Desktop, pois iria agilizar a maneira de como os dados são exibidos ao jogador durante uma partida.

Também seria interessante o desenvolvimento de novas dicas, sugestões de runas a serem utilizadas e ordem de habilidades a serem utilizadas nos campeões.

## **Referências Bibliográficas**

RIOT GAMES. **O que é League of Legends?** Disponível em:

<[https://play.br.leagueoflegends.com/pt\\_BR](https://play.br.leagueoflegends.com/pt_BR)>. Acesso em: 13 out. 2018.

RIOT GAMES. **As funções:** O papel de cada campeão. 2018. Disponível em:

<<https://centraldeaprendizado.br.leagueoflegends.com/videos/funcoes>>. Acesso em: 15 out. 2018.

AQOURS. **Qual o apelo de League of Legends?** [mensagem pessoal] Mensagem recebida por: <CHAMADO N. 37035070, canal de suporte da Riot Games,

- <https://support.riotgames.com/hc/pt-br/>>. em: 11 out. 2018.
- SEULA, Roger Sandro. **Jogando para vencer e jogando para entreter: análise comparativa da performance do pro-player brTT de League of Legends em livestreamings**. 2017. 87 f. Tese (Graduado) - Curso de Publicidade e Propaganda, Universidade Federal do Rio Grande do Sul Faculdade de Biblioteconomia e Comunicação, Porto Alegre, 2018. Disponível em: <<https://www.lume.ufrgs.br/handle/10183/177742>>. Acesso em: 29 out. 2018.
- TIME DO BOOSTRAP. **Bootstrap**. Disponível em: <<https://getbootstrap.com/>>. Acesso em: 23 abr. 2019.
- MICROSOFT. **Introdução à linguagem C# e ao .NET Framework**. 2015. Disponível em: <<https://docs.microsoft.com/pt-br/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>>. Acesso em: 23 abr. 2019.
- MONGODB INC.. **What is MongoDB?** Disponível em: <<https://www.mongodb.com/what-is-mongodb>>. Acesso em: 24 abr. 2019.
- MICROSOFT. **Visual Studio Community**: Everything you need all in one place. Disponível em: <<https://visualstudio.microsoft.com/vs/community/>>. Acesso em: 24 abr. 2019.
- 3T SOFTWARE LABS. **Robo 3T: Native and cross-platform MongoDB manager**. 2017. Disponível em: <<https://robomongo.org/>>. Acesso em: 24 abr. 2019.
- STATCOUNTER. **Browser Market Share Worldwide**. Disponível em: <<http://gs.statcounter.com/>>. Acesso em: 24 abr. 2019.
- TURINE, Marcelo Augusto Santos; MASIERO, Paulo Cesar. **ESPECIFICAÇÃO DE REQUISITOS: UMA INTRODUÇÃO**. 1996. Disponível em: <[https://sistemas.riopomba.ifsudestemg.edu.br/dcc/materiais/2109177910\\_aula%20extra%204%20engenharia\\_de\\_requisitos.pdf](https://sistemas.riopomba.ifsudestemg.edu.br/dcc/materiais/2109177910_aula%20extra%204%20engenharia_de_requisitos.pdf)>. Acesso em: 05 maio 2019.
- JSON.ORG. **Introdução ao JSON**. 1999. Disponível em: <<https://www.json.org/json-pt.html>>. Acesso em: 17 maio 2019.
- GOLART, Jonathan Bernardes; KROEFF, Renata Fischer da Silveira; GAVILLON, Póti Quartiero. **Aprendizagem Colaborativa e Violência Entre Jogadores de League of Legends**. 2017. Disponível em: <<https://www.seer.ufrgs.br/InfEducTeoriaPratica/article/view/67988/41087>>. Acesso em: 21 maio 2019.
- ANDERSON, David. Foreword by David Anderson. In: KNIBERG, Henrik; SKARIN, Mattias. **Kanban and Scrum: making the most of both**. making the most of both. [s. L.]: Lulu.com, 2010. p. 7-10. Disponível em: [http://www.agileinnovation.eu/wordpress/wp-content/uploads/2010/09/KanbanAndScrum\\_MakingTheMostOfBoth.pdf](http://www.agileinnovation.eu/wordpress/wp-content/uploads/2010/09/KanbanAndScrum_MakingTheMostOfBoth.pdf). Acesso em: 04 jun. 2020.
- PERENS, Bruce. **The Open Source Definition**. 2007/2008. Disponível em: <[https://www.researchgate.net/profile/Bruce\\_Perens/publication/200027107\\_Perens\\_Open\\_Source\\_Definition\\_LG\\_26/links/568bdeb408ae16c414a9c549/Perens-Open-Source-Definition-LG-26.pdf](https://www.researchgate.net/profile/Bruce_Perens/publication/200027107_Perens_Open_Source_Definition_LG_26/links/568bdeb408ae16c414a9c549/Perens-Open-Source-Definition-LG-26.pdf)>. Acesso em: 2019 jul. 24.



- ALVES, Alessandro Martins et al. **Logit models for the probability of winning football games**. Pesquisa Operacional, [s.l.], v. 31, n. 3, p.459-465, dez. 2011. FapUNIFESP (SciELO). <http://dx.doi.org/10.1590/s0101-74382011000300003>. Disponível em: <[http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S0101-74382011000300003](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382011000300003)>. Acesso em: 29 jan. 2020.
- HUANG, Thomas; KIM, David; LEUNG, Gregory. **League of Legends Win Predictor**. 2015. Disponível em: <https://thomasythuang.github.io/League-Predictor/abstract.pdf>. Acesso em: 04 jun. 2020.
- YIN, Jihan. **League of Legends: Predicting Wins In Champion Select With Machine Learning**. 2018. Disponível em: <<https://hackernoon.com/league-of-legends-predicting-wins-in-champion-select-with-machine-learning-6496523a7ea7>>. Acesso em: 29 jan. 2020.
- COSTA, Gilcione Nonato. **DOUTOR em Probabilidade da UFMG analisa chances de vitória nos jogos**. Produção de Alterosa Esporte. Realização de Gilcione Nonato Costa. [s.l.]: Alterosa Esporte, 2016. (191 min.), son., color. Entrevista a canal esportivo. Disponível em: <<https://www.youtube.com/watch?v=i6SBOBfZv9g>>. Acesso em: 29 jan. 2020.
- POROFESSOR.GG. **Live Games Stats**: Realtime summoners stats. Disponível em: <<https://porofessor.gg/>>. Acesso em: 04 fev. 2020.
- OUTPLAYED INC. **Frequently Asked Questions**: What is unique about U.GG?. 2019. Disponível em: <<https://u.gg/faq>>. Acesso em: 04 fev. 2020.
- RIOT GAMES INC. **Web APIs**. 2020. Disponível em: <<https://developer.riotgames.com/docs/portal>>. Acesso em: 02 out. 2020.
- SOFTWARE FREEDOM CONSERVANCY. **Git**. Disponível em: <<https://git-scm.com/>>. Acesso em: 19 fev. 2020.
- RIOT GAMES. **Como funciona o gerenciador de partidas?** Disponível em: <<https://centraldeaprendizado.br.leagueoflegends.com/videos/gerenciador-de-partidas>>. Acesso em: 27 fev. 2020.
- UCLA. **COMO SABER SE TEM SMURF OU ELOJOB NO SEU TIME?** 2015. Disponível em: <<https://www.lolpro.com.br/como-saber-se-tem-smurf-ou-eljob-no/>>. Acesso em: 27 fev. 2020.
- RIOT GAMES INC. **POLICIES: PLEASE DON'T**. 2020. Disponível em: <<https://developer.riotgames.com/policies/general>>. Acesso em: 27 fev. 2020.
- FLORES, Azul Romo. **To Conquer Or To Kill?**: a qualitative study on game mechanics and player experience in moba games. A qualitative study on game mechanics and player experience in MOBA games. 2018. Disponível em: <http://www.diva-portal.org/smash/get/diva2:1304821/FULLTEXT01.pdf>. Acesso em: 16 jun. 2020.

- COLEMAN, Robert. **Good Game, Well Played:** The toxic effect of competitive video games. The Toxic Effect of Competitive Video Games. 2019. Disponível em: <http://147.226.7.105/bitstream/handle/123456789/201987/2019ColemanRobert-combined.pdf?sequence=1&isAllowed=y>. Acesso em: 04 jun. 2020.
- WAZLAWICK, Raul. **Engenharia de software:** conceitos e práticas. 2. ed. Florianópolis: Elsevier Editora Ltda., 2019. 320 p.
- KIM, Jooyeon; KEEGAN, Brian C.; PARK, Sungjoon; OH, Alice. **The Proficiency-Congruency Dilemma:** virtual team design and performance in multiplayer online games. Virtual Team Design and Performance in Multiplayer Online Games. 2015. Disponível em: <https://arxiv.org/pdf/1512.08321.pdf>. Acesso em: 16 jun. 2020.

## APÊNDICE A – Detalhamento dos casos de uso

<b>Nome do Caso de Uso:</b> Inserir usuário e região do jogo
<b>Ator Principal:</b> Jogador
<b>Atores Secundários:</b>
<b>Resumo:</b> Inserção de dados do jogador para possibilitar a identificação e cálculo de estatísticas.
<b>Pré Cond:</b> Nome de jogador existir na região selecionada.
<b>Pós-Cond:</b> -
<b>Ações do Ator:</b>
1.Inserir nome de jogador;
2.Selecionar região;
3.Selecionar função desejada;
<b>Ações do Sistema:</b>
1.Verificar se nome de jogador existe na região selecionada;
2.Direcionar jogador para a função desejada;
<b>Nome do Caso de Uso:</b> Consultar partida atual
<b>Ator Principal:</b> Jogador
<b>Atores Secundários:</b>
<b>Resumo:</b> Listar informações que auxiliem o usuário a vencer a partida atual.
<b>Pré Cond:</b> Jogador deve encontrar-se em uma partida.
<b>Pós-Cond:</b> -
<b>Ações do Ator:</b>
1.Observar as informações;
<b>Ações do Sistema:</b>
1.Listar as informações que irão auxiliar o usuário;
<b>Nome do Caso de Uso:</b> Buscar histórico de partidas
<b>Ator Principal:</b> Jogador
<b>Atores Secundários:</b>
<b>Resumo:</b> Exibição do histórico de partidas, seus detalhes e algumas estatísticas.

<b>Pré Cond:</b> Jogador ter jogado alguma partida nas últimas 4 temporadas <sup>17</sup> .
<b>Pós-Cond:</b> -
<b>Ações do Ator:</b>
1.Observar o histórico de partidas, seus detalhes e algumas estatísticas;
<b>Ações do Sistema:</b>
1.Exibir histórico de partidas, seus detalhes e algumas estatísticas;
<b>Nome do Caso de Uso:</b> Ver detalhes da partida jogada
<b>Ator Principal:</b> Jogador
<b>Atores Secundários:</b>
<b>Resumo:</b> Exibição de alguns detalhes da partida, especificados nos requisitos.
<b>Pré Cond:</b> Usuário acessar a tela de partidas jogadas.
<b>Pós-Cond:</b> -
<b>Ações do Ator:</b>
1.Visualizar detalhes da partida, que estão descritos nos requisitos;
<b>Ações do Sistema:</b>
1.Exibir detalhes da partida;
<b>Nome do Caso de Uso:</b> Ver estatísticas gerais do jogador
<b>Ator Principal:</b> Jogador
<b>Atores Secundários:</b>
<b>Resumo:</b> Manutenção dos dados das sepulturas.
<b>Pré Cond:</b> Usuário acessar a tela de partidas jogadas.
<b>Pós-Cond:</b> -
<b>Ações do Ator:</b>
1.Visualizar estatísticas;
<b>Ações do Sistema:</b>
1.Exibir as estatísticas especificadas nos requisitos;

<sup>17</sup> Temporada é a época em que as partidas classificatórias de *elo* estão disponíveis, geralmente acontecem duas temporadas por ano.

## APÊNDICE B – Detalhamento dos resultados dos testes em partidas

Aqui temos o detalhamento de cada análise feita pelo Fajlol para cada jogador. Estes são os mesmos dados da Tabela 3 porém não estão agrupados. Representando cada análise com cada jogador com quem a ferramenta foi testada.

**Tabela 4 – Detalhamento dos resultados dos testes em partidas.**

Partida	Jogador	Jogou com campeão recomendado?	Venceu?	Ferramenta acertou?
1	1	✓		
2	1	✓	✓	✓
3	1	✓		
4	1	✓	✓	✓
1	2			
2	2		✓	✓
3	2	✓		
4	2	✓	✓	✓
1	3	✓		
2	3	✓	✓	✓
3	3			
4	3	✓	✓	✓
Total		9	6	6