Consumo e análise de dados de API para ajudar em tomadas de decisão num ambiente competitivo

Lucas Cardoso de Assis[1], Lucas Oliveira Coelho[2], Fabrício Henrique Neto[3]

[1]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)
Ribeirão Preto, SP – Brasil

[2]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)
Ribeirão Preto, SP – Brasil

[3]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)
Ribeirão Preto, SP – Brasil

lucasca972@gmail.com, l.coelhooliveira@gmail.com,
fabricio.henrique@fatec.sp.gov.br

Resumo. *O objetivo principal do projeto é o consumo de dados crus e complexos para fornecer informações concisas e simples para o usuário final, se baseando no cenário competitivo do jogo League of Legends, analisando os melhores jogadores brasileiros e apresentando informações que possam ser úteis sobre como eles jogam.*

Abstract. *The main goal of the project is the consumption of raw and complex data to give simple and concise information to the final user, using as parameter the competitive scene in the game League of Legends, analysing the best brazilian players in the game and presenting information that can be useful about how they play.*

1. Introdução

Nos últimos anos o mundo presenciou o fenômeno dos *E-sports* (esportes eletrônicos). Em 2020 estimou-se que 2,6 bilhões de pessoas jogavam algum tipo de vídeo game, sendo 266 milhões destes presentes na América Latina. Para eventos de E-sports estima-se 646 milhões de espectadores para 2020, demonstrando um crescimento percentual de 10,4% em relação ao ano anterior [Newzoo 2019].

O *League of Legends* (LoL), desenvolvido pela *Riot Games*, tornou-se um dos *E-sports* do momento. No campeonato mundial de 2020 o pico de espectadores simultâneos assistindo às transmissões online contou com 1,168 milhões de pessoas, somando um número de horas assistidas na fase de grupos de aproximadamente 29,664 milhões [SportsProMedia 2020].

A Riot Games fornece dados sobre jogadores e partidas, além de diversos tipos de informações por meio de uma Interface de programação de aplicações (Application Programming Interfaces - API), descrita na seção 2.1, permitindo a coleta de um conjunto de dados complexos que quando analisados geram informações específicas no escopo do jogo.

Considerando o crescente cenário dos E-sports, o objetivo desse trabalho, com foco nas competições do LoL, pretende-se fornecer estatísticas e informações que auxiliem o usuário final a formular sua melhor estratégia tendo como base, dados de partidas dos melhores jogadores do ranking atual. Apresentaremos todas as informações em uma interface *user-friendly*, fazendo uso de tecnologias bem conceituadas no mercado de trabalho (ver Figura 3), descritas na seção 2.

2 Tecnologias Utilizadas

A partir de uma conta de desenvolvedor (*Riot Games*) e uma chave de acesso às suas **API**s (seção 2.1), que possuem o limite de até 20 requisições por segundo ou até 100 requisições a cada 2 minutos, temos acesso aos dados das partidas.

Para realizar esse acesso utilizou-se o NodeJs (seção 2.3) como Back-end. Os dados fornecidos em formato **JSON**s (seção 2.2) passaram por um tratamento antes de serem armazenados no banco de dados.

No banco relacional **PostgreSQL** (seção 2.4) os dados são cruzados e assim obtidas as informações apresentadas para o usuário final, criando os **Endpoints** (seção 2.1) que servem a aplicação no Front-end em **React** (seção 2.6).

2.1 Rest Api

API, *Application Programming Interface*, é uma programação padronizada de rotina [CanalTech 2017], normalmente disponível por um *link Web*, seja uma API aberta para todos ou apenas com chaves de acesso específicas. APIs comumente têm sido usadas para passagem de dados simples pela Web, só é preciso ter o link correto para conseguir a informação necessária e esse link correto é o Endpoint.

REST, REpresentational State Transfer, é um estilo de arquitetura web, basicamente ela garante que a passagem de dados e a comunicação entre as partes que trabalham em conjunto para página web funcionar seja feita de forma segura e correta, quando os princípios da arquitetura REST são aplicados nos dados fornecidos pela API ela é considerada RESTful [Restful API 2018].

2.2 Json

JSON, *JavaScript Object Notation*, é um dos formatos mais utilizados para passagem de dados na *Web*, uma das vantagens do JSON é que ele é facilmente lido tanto pelo usuário humano quanto pelas linguagens de programação, justamente por ter uma estrutura simples e prática [Dr. Derek Austin 2020], segue exemplo de JSON:

```
[
    {
        "summonername": "Hide on bush",
        "profileicon": 6,
        "rankposition": 1,
        "wins": 1109,
        "losses": 939,
        "leaguepoints": 1542
    }
]
```

Figura 1: Estatísticas do jogador top 1 do Ranking.

2.3 NodeJs

Segundo o próprio site oficial do Node, ele é considerado uma solução da linguagem de programação JavaScript assíncrona e impulsionada por eventos, assíncrona porque ele te dá a opção de fazer vários “pontos de processamento” ao mesmo tempo, mesmo que o anterior não tenha terminado, seja com o *async/await* ou com alguma função que já trabalhe dessa forma internamente, assim fornecendo mais opções para os desenvolvedores trabalharem com as aplicações, quando um bloco de código é executado de forma assíncrona e o sistema “espera” a resposta daquele processamento, essa resposta é chamada de *Promise* [Lucas Santos 2019]. E é considerado impulsionado em eventos por estar o tempo todo “escutando” se algum gatilho foi disparado o pedindo para fazer algum tipo de processamento.

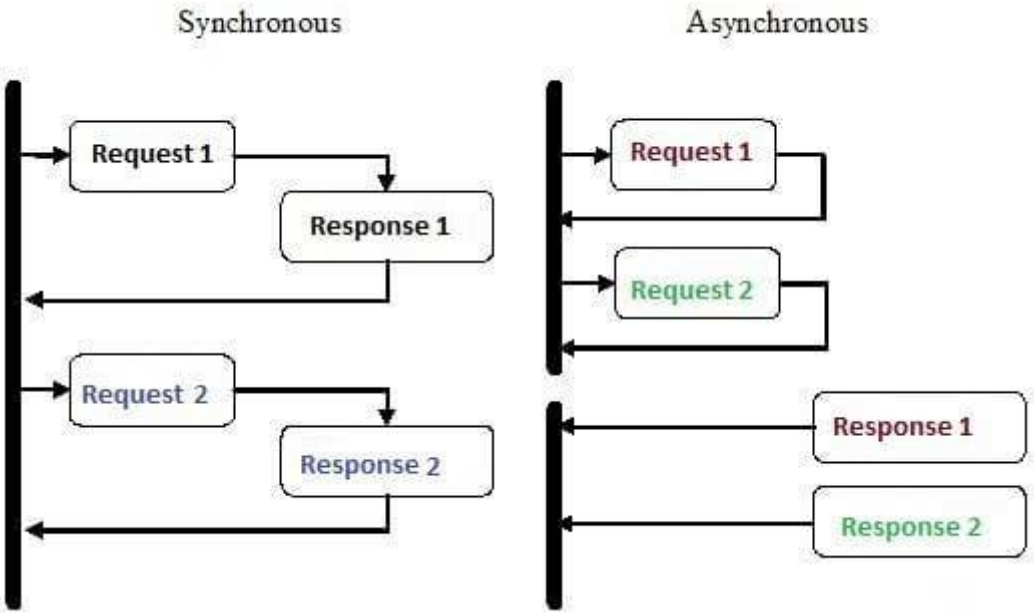

Figura 2: Comparação de Processamento Síncrono e Assíncrono. [Luiz Duarte 2020]

O Node trás várias ferramentas interessantes como seus módulos, cada um com sua funcionalidade sendo escolhidos conforme a necessidade, neste trabalho foram usados o *axios* pra consumir os dados da API, *express* para construir a aplicação web e criar a API e o *nodemon* para facilitar os testes locais.

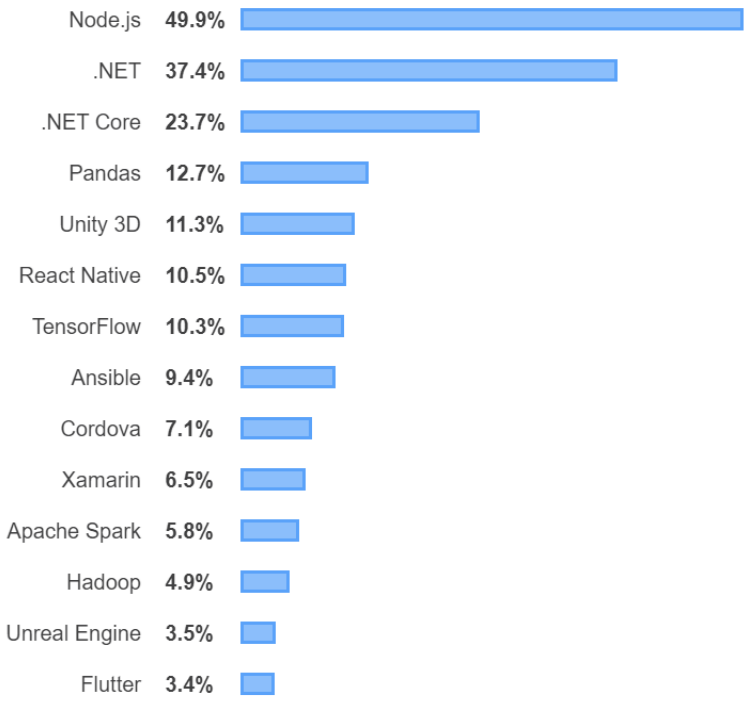

Figura 3: Pesquisa de popularidade realizada pelo Stack Overflow em 2019 [Stack Overflow 2019]

2.4 PostgreSQL

Um banco de dados é um conjunto de informações propriamente estruturado e organizado, normalmente no padrão de tabelas interligadas com uma lógica por trás, para facilitar no controle e nas consultas de informações, a maioria dos bancos de dados usa a linguagem SQL, Structured Query Language, para interagir com os dados do banco, também é necessário um SGBD, Sistema de Gerenciamento de Banco de Dados, para fazer essa ponte entre o usuário que vai interagir com o banco e as próprias informações do mesmo [Oracle].

É necessário que o banco de dados forneça algumas propriedades para se provar um banco de qualidade, normalmente o básico é seguir a sigla ACID: Atomicidade, Consistência, Isolamento e Durabilidade, tudo para que os dados estejam de fato seguros e que não haverão problemas no funcionamento, é o mínimo que se espera de um banco bem conceituado no mercado.

O banco de dados escolhido para o projeto foi o PostgreSQL por estar a vários anos no mercado e ser uma solução que se mostrou confiável e com boa performance [PostgreSQL], ele funciona da forma que foi dita acima, com a estrutura de tabelas interligadas, SGBD próprio e linguagem SQL.

2.5 React

React é uma biblioteca em JavaScript focada em criar interfaces para o usuário, no caso desse projeto o objetivo é criar o *Front-end* da aplicação *Web*. O React surgiu do *Facebook* em 2011, se demonstrou uma alternativa promissora para interfaces web por ser declarativa e trabalhar com seus componentes reutilizáveis para formar a página, e segundo os dados é a solução mais popular em número de downloads no npm, gerenciador de pacotes JavaScript.

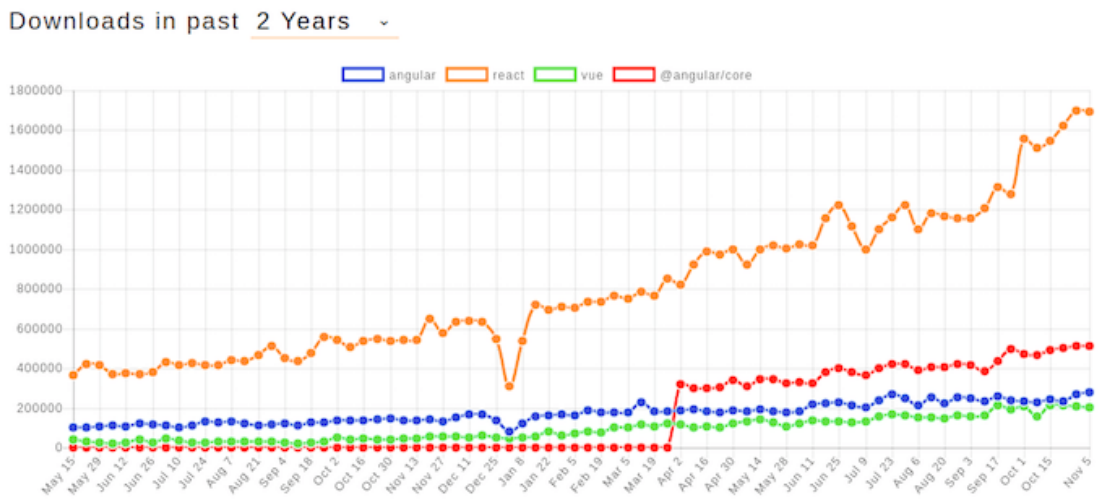

Figura 4: Principais soluções para interface web e seu número de Downloads (NPM). [Jens Neuhaus 2017]

O React trabalha com uma sintaxe chamada JSX, formando uma pseudo-linguagem da mescla de JavaScript e de XML, isso também facilita o funcionamento dos componentes reativos [Davi Ferreira 2013].

Outro conceito importante de React é o Virtual DOM, *Document Object Model*, o DOM comum é, independente da linguagem de programação ou do navegador, por consenso usado para representação e interação dos objetos HTML da página Web interpretado pelo próprio navegador. Já o Virtual DOM no React é usado para simular esse ambiente virtualmente para depois ser sincronizada com o DOM "real" [React].

3. Procedimentos Metodológicos

3.1 Consumo da API

Para o consumo da API foi necessário criar uma conta de desenvolvedor na Riot Games em https://developer.riotgames.com/, com essa conta gerar uma chave de acesso, liberando até 20 "consumos" de API por segundo ou 100 "consumos" a cada 2 minutos, à partir disso foi preciso estudar a documentação da própria desenvolvedora explicando como funcionam suas APIs, que tipo de dados elas retornam e quais dados algumas delas precisam para retornar alguma informação, depois foi traçado o caminho que seria necessário entre as APIs:

a) Descobrir os 200 melhores do *Ranking* e pegar seus "SummonerIds".
b) Usar o "SummonerIds" pra pegar o "AccountId".
c) Usar o "AccountId" pra ver as últimas partidas do jogador.
d) Usar o Id das partidas para ver os detalhes da mesma.
e) Filtrar os dados.

Já para lidar com a limitação de requisições nas APIs foi utilizado o TeemoJS, um módulo que permite tentar consumir a API novamente de onde deu o erro de limite de requisições da última vez, foi necessário trabalhar em cima dessa limitação porque o número de requests passa tranquilamente do limite estipulado pela desenvolvedora do jogo, totalizando em 801 *Requests*, observe:

f) 1 *Request* para descobrir os 200 melhores jogadores.
g) 200 *Requests*, 1 para cada jogador pra pegar o AccountId de cada um.
h) 200 *Requests*, para pegar o histórico de partidas de cada jogador.
i) 400 *Requests*, para ver os detalhes de 2 partidas de cada jogador.

3.2 Backend com Node.js

No Backend em Node que os requests das APIs foram feitos e controlados para não exceder o limite por tempo, a opção de trabalhar com a **programação assíncrona** e com as **Promises** facilitou esse processo.

Para fazer os requests foi usado o **Axios**, o fato de os dados virem no formato JSON facilita muito trabalhar como se fossem objetos JavaScript, por isso ao percorrer os dados com a função *map()*, que tem justamente esse objetivo, é possível manipular esses objetos da forma necessária. Assim, usando as funções assíncronas, é necessário mapear qual parte dos dados nos interessa e fazer o caminho entre as APIs, no fim do processo cada jogador tem seus dados corretamente agrupados para assim finalmente serem inseridos no banco de dados.

O limite por tempo do consumo das APIs foi resolvido usando o *TeemoJS*, que funciona inserindo pequenas pausas de tempo quando aparece um problema ao fazer um request na API, fazendo o código precisar de mais tempo para terminar de executar, mas também evitando problemas.

3.3 Banco de dados PostgreSQL

Após consumir as diversas APIs do *League of Legends* e gerar os dados estruturados, o *Back-end* em NodeJS salva esses dados nas tabelas do banco PostgreSQL, como a API do *League of Legends* tem um limite baixo de requisições por minuto, as informações são salvas em tabelas jogador por jogador para depois realizar as consultas e análises, o banco foi estruturado da seguinte forma:

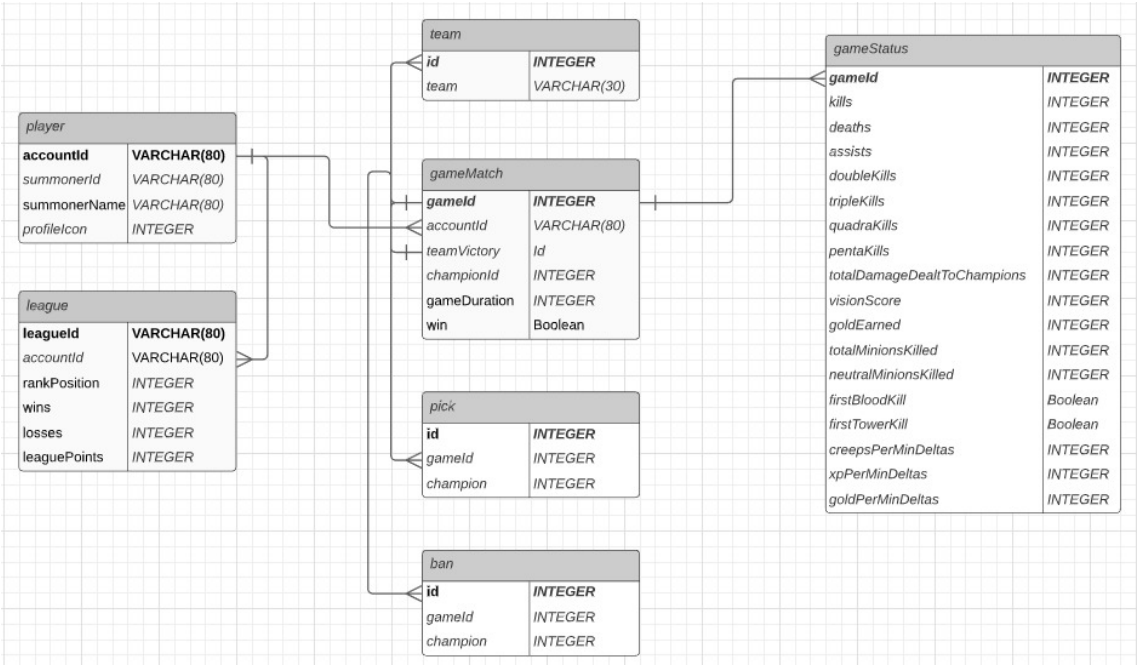

Figura 5: Detalhamento das tabelas e suas colunas.

3.4 Frontend com React

A biblioteca React se encarrega em formar toda a interface e fazer com que os elementos da página façam sentido, como por exemplo, apresentar os dados de "maior taxa de escolha de campeões" fazendo com que aquela estatística que chega por meio de um

arquivo JSON seja corretamente interpretada e mostrada para o usuário de uma forma agradável, inclusive formando gráficos.

Com a biblioteca Material UI do React foi desenvolvida a base do Front-end, o Axios entra novamente para consumir os dados do banco de dados e os apresentar em tela. Também foi usado o react-router-dom para criar e organizar as rotas de URL da aplicação final.

Os gráficos criados com a biblioteca “dx-react-chart-material-ui” fornecem uma visualização simples e prática para o usuário, as estatísticas que comportam gráficos para visualização de informações tem uma nova cara.

4. Resultados E Discussões

Ao aplicar as tecnologias e estratégias citadas acima foi possível alcançar resultados que poderão servir de guia para possíveis jogadores que se interessarem em buscar informações para melhor performance nas suas competições.

Na Figura 6, pode-se ver a relação dos melhores jogadores do ranking que implica nos jogadores que possuem os maiores pontos de liga (PDL), que determinam sua posição no *Ranking*, junto com seu número de vitórias e derrotas em partidas.

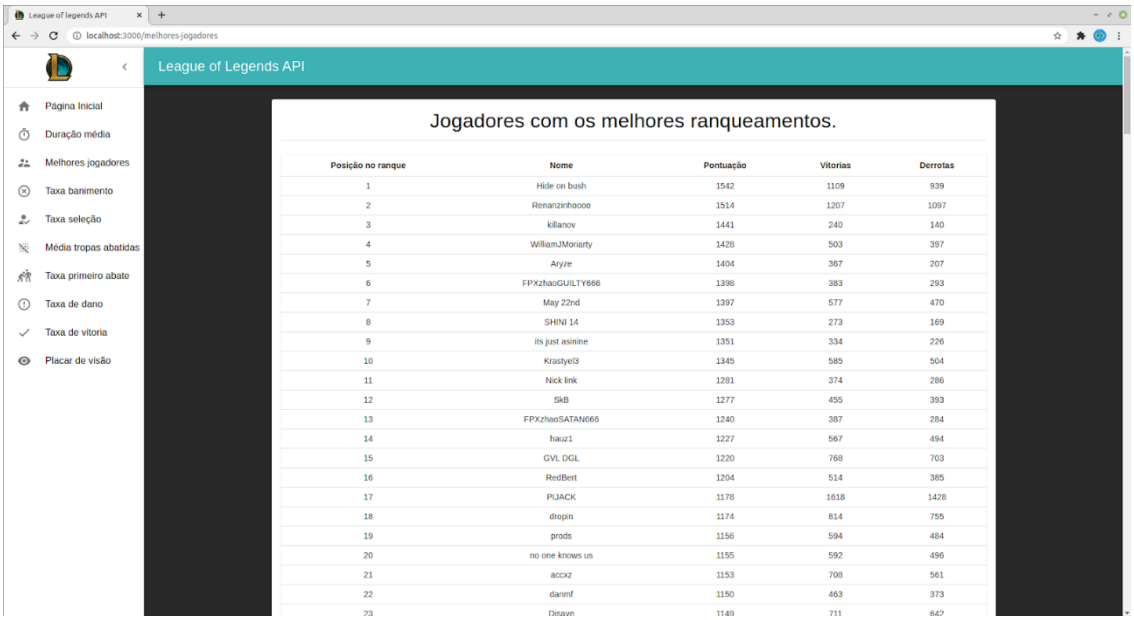

Figura 6: Melhores jogadores do Ranking.

A Figura 7 apresenta tempo médio de todas as partidas analisadas no projeto.

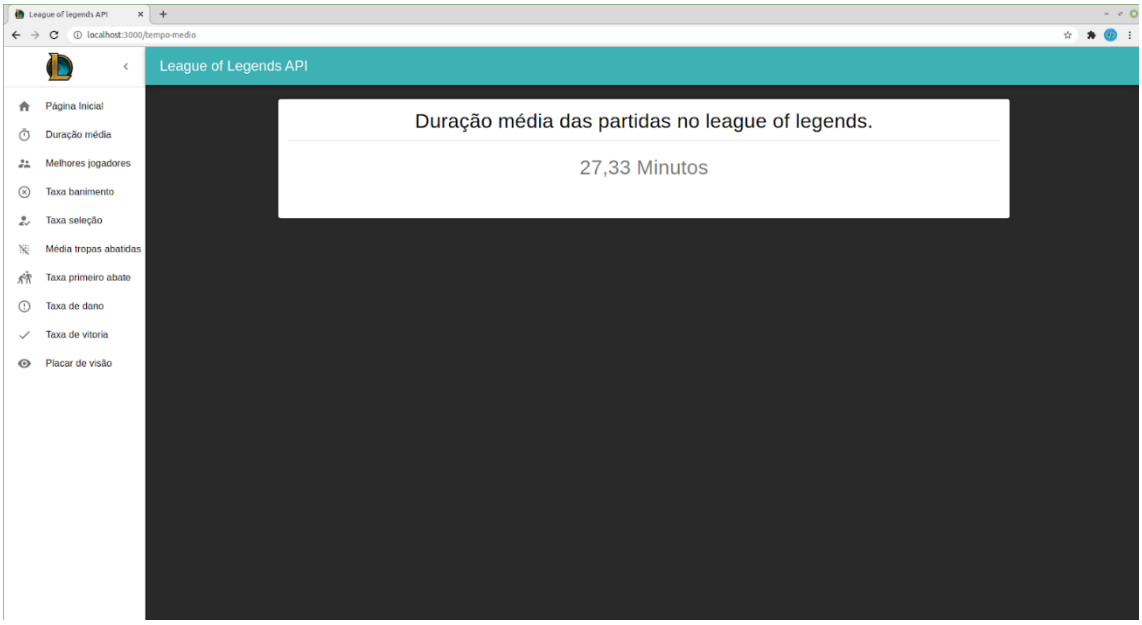

Figura 7: Tempo médio de partidas.

Na Figura 8, apresentamos os dez campeões com a maior taxa de seleção com gráficos e logo abaixo a lista completa de todos os personagens que foram selecionados nas partidas analisadas.

Figura 8: Taxa de seleção dos campeões.

Na Figura 9, temos a relação com maior taxa de banimento, ou seja, são os dez campeões com maior probabilidade de serem vetados em uma partida.

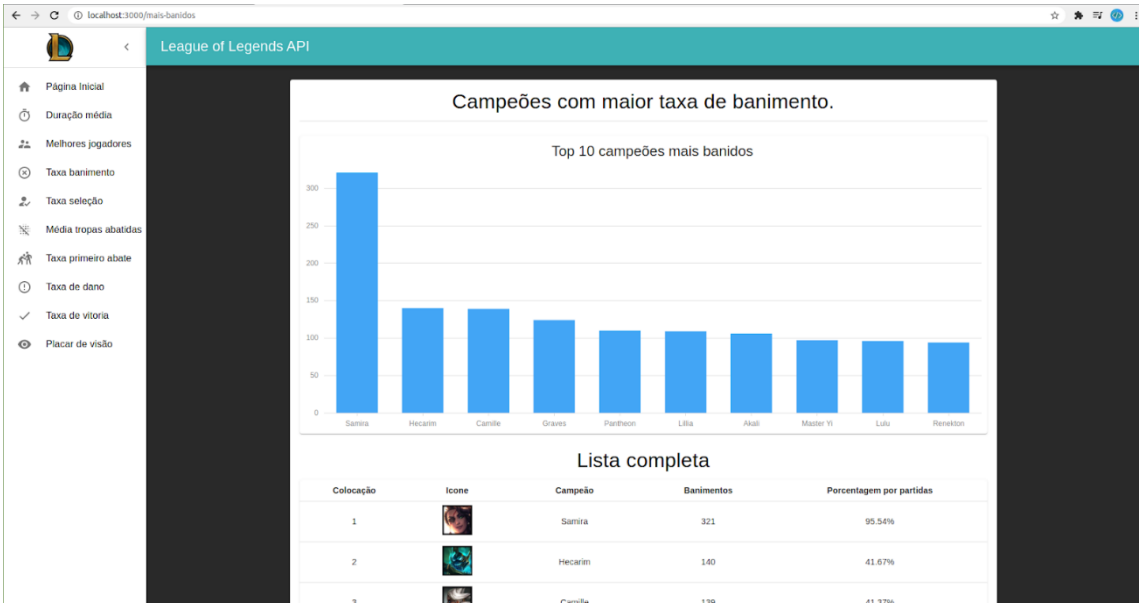

Figura 9: Taxa de banimento dos campeões.

Na Figura 10, podemos ver a listagem de campeões com mais primeiros abates em partidas, o que favorece estratégias que focam mais no início do jogo.

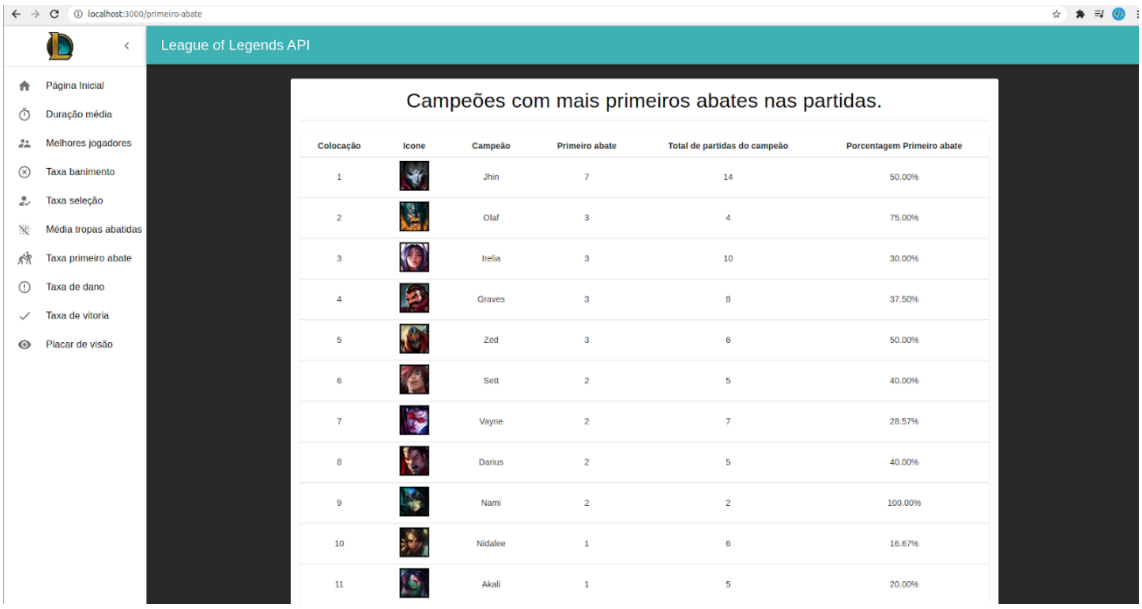

Figura 10: Campeões com mais "primeiros abates" nas partidas.

Na Figura 11, apresentamos a relação de campeões com maior dano por partida, indicando quem são os personagens com maior potencial de fazer a diferença numa estratégia em times bem estruturados.

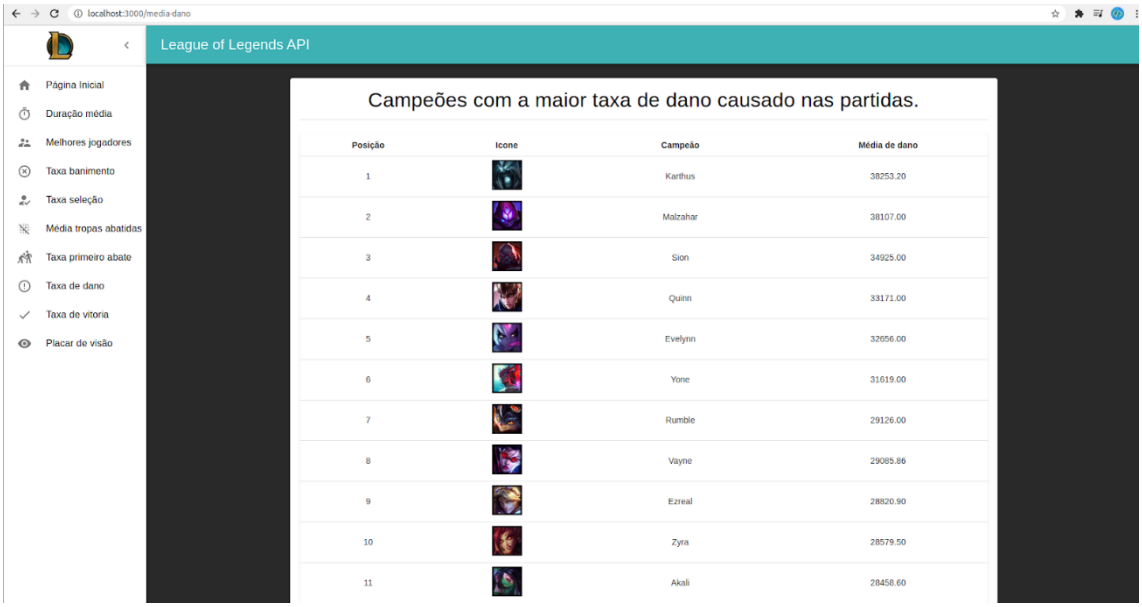

Figura 11: Campeões com maior dano por partida.

Figura 12, apresentamos os campeões com maior número de vitórias em partidas.

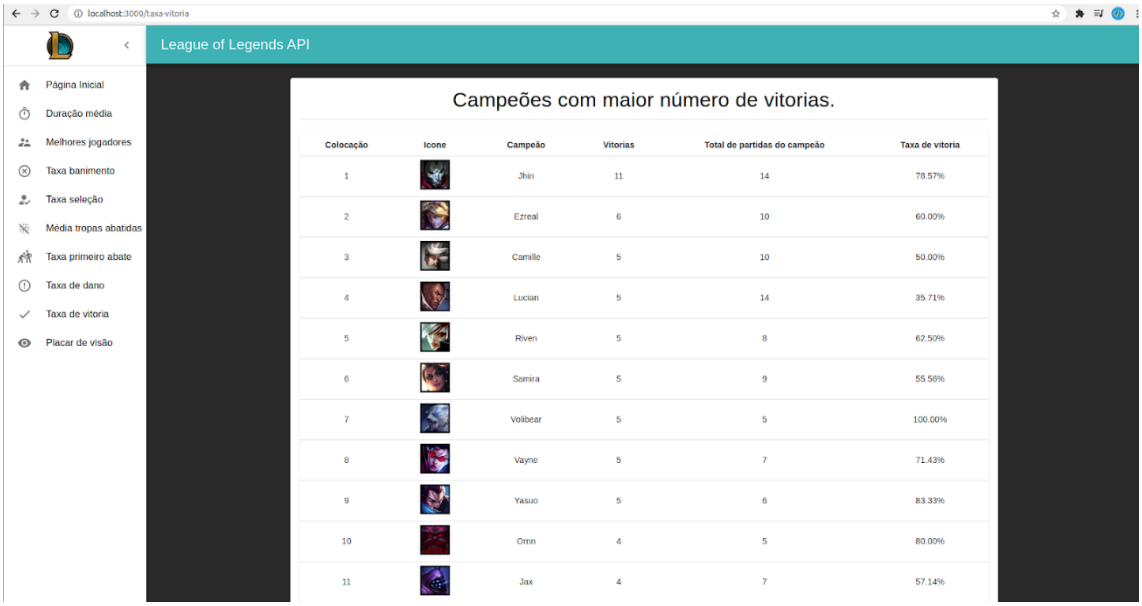

Figura 12: Campeões com a maior taxa de vitória.

Figura 13, apresentamos a média de abate de tropas neutras por partida, o abate de tropas neutras é importante porque gera dinheiro para os personagens e dinheiro pode ser usar para comprar itens que melhoram os atributos do personagem.

Figura 13: Média de abate de tropas neutras por campeão por partida.

Figura 14, apresentamos a média de pontuação de visão por partida, a visão é um ponto importante para que seu time tenha controle de tudo que acontece no mapa e o time adversário não.

Figura 14: Média de Pontuação de visão por partida.

Todas essas informações usando os dados das partidas dos melhores jogadores do *Ranking*.

5. Conclusão

Ao finalizar o projeto é notável que ao aplicar as técnicas e tecnologias citadas é possível apresentar informações sobre os melhores jogadores do *Ranking* do jogo *League of Legends*, da desenvolvedora Riot Games, de uma forma simples usando os dados da API da própria desenvolvedora, formando uma aplicação Web completa e inclusive prever que um projeto mais aprofundado conseguiria fornecer ainda mais dados dos melhores jogadores para quem tem interesse em melhorar sua performance se baseando nisso.

6. Referências

C Kemp; P R Pienaar; D E Rae. **Brace yourselves: esports is coming.** 2020. Disponível em: <http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S1015-51632020000100007&lang=pt>. Acesso em 08 de novembro de 2020.

Newzoo. **Newzoo Global Esports Market Report 2019 | Light Version.** 2019. Disponível em:<https://newzoo.com/insights/trend-reports/newzoo-global-esports-market-report-2019-light-version>. Acesso em 08 de novembro de 2020.

Tom Wijman. **Global Game Revenues Up an Extra $15 Billion This Year as Engagement Skyrockets**. 2020. Disponível em: <https://newzoo.com/insights/articles/game-engagement-during-covid-pandemic-adds-15-billion-to-global-games-market-revenue-forecast/>. Acesso em 28 de novembro de 2020.

Ed Dixon. **2020 LoL World Championship first stage audience peaks at 1.168m**. 2020. Disponível em: <https://www.sportspromedia.com/news/league-of-legends-world-championship-2020-first-stage-audience-viewership> Acesso em 28 de novembro de 2020.

NPD. **The NPD Group: U.S. Consumer Spend on Video Game Products Continues to Break Records**. 2020. Disponível em: <https://www.npd.com/wps/portal/npd/us/news/press-releases/2020/the-npd-group-us-consumer-spend-on-video-game-products-continues-to-break-records/#.XzFIUzekrEY.twitter> Acesso em 28 de novembro de 2020.

RestfulApi. **What is REST**. Disponível em:<https://restfulapi.net/> Acesso em 28 de novembro de 2020.

Equipe TOTVS. **Arquitetura REST: Saiba o que é e seus diferenciais**. Disponível em: <https://www.totvs.com/blog/developers/rest/> Acesso em 28 de novembro de 2020.

Canaltech. **O que é API?**. Disponível em: <https://canaltech.com.br/software/o-que-e-

api/> Acesso em 28 de novembro de 2020.

RestfulApi. **What is JSON**. Disponível em: <https://restfulapi.net/introduction-to-json/> Acesso em 28 de novembro de 2020.

AUSTIN, Derek. **What is JSON Used For in JavaScript Programming?**. 2020. Disponível em: <https://medium.com/swlh/what-is-json-used-for-in-javascript-programming-9d71284359a9> Acesso em 28 de novembro de 2020.

Luiz Duarte. **Processamento assíncrono de tarefas com filas no RabbitMQ e Node.js**. 2020. Disponível em: <https://www.luiztools.com.br/post/processamento-assincrono-de-tarefas-com-filas-no-rabbitmq-e-node-js/> Acesso em 28 de novembro de 2020.

Lucas Santos. **Entendendo Promises de uma vez por todas**. 2019. Disponível em: <https://medium.com/trainingcenter/entendendo-promises-de-uma-vez-por-todas-32442ec725c2> Acesso em 28 de novembro de 2020.

NodeJS. **About Node.js®**. Disponível em: <https://nodejs.org/en/about/> Acesso em 28 de novembro de 2020.

Smartbear. **API Endpoints - What Are They? Why Do They Matter?**. Disponível em: <https://smartbear.com/learn/performance-monitoring/api-endpoints/> Acesso em 28 de novembro de 2020.

NPM. **TeemoJS**. 2020. Disponível em: <https://www.npmjs.com/package/teemojs> Acesso em 28 de novembro de 2020.

ReactJs. **ReactJS**. Disponível em: <https://pt-br.reactjs.org/> Acesso em 28 de novembro de 2020.

Newzoo. **Key Numbers**. Disponível em: <https://newzoo.com/key-numbers> Acesso em 28 de novembro de 2020.

Davi Ferreira. **React: JavaScript reativo**. 2013. Disponível em: <https://tableless.com.br/react-javascript-reativo/> Acesso em 28 de novembro de 2020.

Udacity Brasil. **React: o que é e como funciona essa ferramenta?**. 2018. Disponível em: <https://tableless.com.br/react-o-que-e-e-como-funciona-essa-ferramenta/> Acesso em 28 de novembro de 2020.

Riot Games. **Developer Riot Games**. Disponível em: <https://developer.riotgames.com/> Acesso em 28 de novembro de 2020.

Jens Neuhaus. **Angular vs. React vs. Vue: A 2017 comparison**. 2017. Disponível em: <https://medium.com/pixelpassion/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176> Acesso em 28 de novembro de 2020.

PostgreSQL. **PostgreSQL: The World's Most Advanced Open Source Relational Database**. Disponível em: <https://www.postgresql.org/> Acesso em 28 de novembro de 2020.

Oracle. **O Que É um Banco de Dados?**. Disponível em: <https://www.oracle.com/br/database/what-is-database/> Acesso em 28 de novembro de 2020.

Stack Overflow. **Developer Survey Results 2019**. 2019. Disponível em: <https://insights.stackoverflow.com/survey/2019#technology> Acesso em 28 de novembro de 2020.