Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

API de Sistemas *Matchmaking*: Um Estudo Exploratório Orientado ao Paradigma de Sistemas Multiagentes

Autores: Pedro Henrique Andrade Féo
e Saleh Nazih Abdel Kader
Orientador: Profa. Dra. Milene Serrano

Brasília, DF
2022

Pedro Henrique Andrade Féo
e Saleh Nazih Abdel Kader

API de Sistemas *Matchmaking*: Um Estudo Exploratório Orientado ao Paradigma de Sistemas Multiagentes

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Profa. Dra. Milene Serrano

Coorientador: Prof. Dr. Maurício Serrano

Brasília, DF

2022

Pedro Henrique Andrade Féo
e Saleh Nazih Abdel Kader

API de Sistemas *Matchmaking*: Um Estudo Exploratório Orientado ao Paradigma de Sistemas Multiagentes/ Pedro Henrique Andrade Féo
e Saleh Nazih Abdel Kader. – Brasília, DF, 2022-

91 p. : il. (algumas color.) ; 30 cm.

Orientador: Profa. Dra. Milene Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2022.

1. Matchmaking. 2. Multiagentes. I. Profa. Dra. Milene Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. API de Sistemas *Matchmaking*: Um Estudo Exploratório Orientado ao Paradigma de Sistemas Multiagentes

CDU 02:141:005.6

Pedro Henrique Andrade Féo
e Saleh Nazih Abdel Kader

API de Sistemas *Matchmaking*: Um Estudo Exploratório Orientado ao Paradigma de Sistemas Multiagentes

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 26 de Setembro de 2022:

Profa. Dra. Milene Serrano
Orientador

Prof. Dr. Maurício Serrano
Coorientador

Prof. Dr. Fernando William Cruz
Examinador 1

Prof. Dr. Bruno César Ribas
Examinador 2

Brasília, DF
2022

Agradecimentos

Primeiramente, agradecemos a Deus, por ter nos dado forças nos momentos mais difíceis, e por ter nos conduzidos até aqui.

Agradecemos aos nossos Professores Orientadores, Profa. Dra. Milene Serrano e Prof. Dr. Maurício Serrano, por seus ensinamentos, por suas orientações, e por toda paciência que tiveram conosco ao longo do trabalho. Vocês são inspirações para nós.

Agradecemos às nossas famílias, por nos apoiarem nos momentos mais difíceis, e acreditarem em nós ao longo de nossa graduação.

Agradecemos aos nossos amigos, por serem quem são, por todo apoio, tornando a nossa jornada, ao longo da graduação, mais descontraída e divertida.

A todos esses, nosso mais sincero obrigado.

"Às vezes a vida é como este túnel escuro, nem sempre se vê a luz no fim do túnel, mas se você continuar em frente, você chegará a um lugar melhor"
(Iroh, Avatar: A Lenda de Aang)

Resumo

Matchmaking é o processo de busca e organização de jogadores, dentro de um jogo *online*, afim de se construir a melhor experiência possível dentro de um jogo. Apesar de ser uma problemática muito abordada dentro do mundo de jogos *multiplayer online*, a implementação dos Sistemas *Matchmaking* varia para a maioria dos jogos, seja por regras específicas de cada um, ou por atributos diferentes a serem considerados para se estabelecer o que é uma boa experiência. Através da utilização do modelo conceitual de Multiagentes, o trabalho propõe o uso de cognição e entidades inteligentes para lidar com essa problemática. Os Sistemas Multiagentes se mostram pertinentes nessa problemática por evidenciarem capacidade de comunicação e autonomia, criando a possibilidade de que os Agentes possam negociar entre si para formar os melhores grupos possíveis. Uma das intenções desse trabalho consistiu em avaliar se a utilização do Paradigma de Sistemas Multiagentes comportou-se de forma aderente para a implementação de Sistemas *Matchmaking*. No intuito de se ter uma visão mais concreta, procurou-se implementar uma *Application Programming Interface* (API) de Sistemas *Matchmaking* genérica, ou seja, de fácil utilização em qualquer jogo.

Palavras-chave: jogos *online*. jogos digitais. *matchmaking*. multiagentes. API

Abstract

Matchmaking is the process of searching and organizing players inside an online multiplayer game, with the goal of building the best possible experience in a game. Despite being a commonly approached topic in the online multiplayer gaming community, the implementation of Matchmaking Systems differ in every game. This is caused by the different rules and attributes that a game considers to impact the game experience. By utilizing the Multiagent conceptual model, this work proposes the use of cognition and smart entities to deal with this problem. Multiagent Systems are relevant in this scenario for having the ability of being autonomous and communicative, enabling Agents to negotiate between themselves to achieve the best possible matchs. One of the intentions of this work consisted in asessing the use of Multiagent Systems on the implementation of Matchmaking Systems. With the objective of having a more concrete vision, an Application Programming Interface (API) for generic Matchmaking systems, which could be easily used in any game, were implemented.

Key-words: online games. digital games. matchmaking. multiagents. API.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

MOBA	*Multiplayer Online Battle Arenas*
LOL	*League of Legends*
SMA	Sistema Multiagentes
GDBL	Aprendizado Baseado no Desenvolvimento de Jogos
WIP	*Work in Progress* - Trabalho em Progresso
GaaS	*Game as a Service* - Jogo como Serviço
API	*Application Programming Interface*
CFP	*Call for Proposals*

Sumário

1 Introdução

O presente capítulo visa apresentar as considerações iniciais acerca do presente Trabalho de Conclusão de Curso. Em um primeiro momento, será abordada a Contextualização, focada em apresentar o domínio de interesse deste trabalho (i.e. Domínio de Jogos), bem como o seu processo de interesse, denominado por *Matchmaking*. Será acordada também a Justificativa para a realização deste Trabalho, buscando esclarecer as contribuições ao Domínio de Jogos e ao processo de *Matchmaking*, usando como orientação o Paradigma de Programação Sistemas Multiagentes. Na sequência, serão descritos Questão de Pesquisa, respondida ao final do trabalho, e os Objetivos. Por fim, será conferida a Organização da Monografia.

1.1 Contextualização

Quando se pronuncia a palavra jogo, cada um pode entendê-la de modo diferente, de acordo com KISHIMOTO (1997). Considerando a origem do termo, o latim cobre todo o contexto do jogo com uma única palavra: "*ludus*". Entretanto, essa palavra foi substituída por seu derivado chamado de "*jocus*", cujo sentido é gracejar, troçar, sendo ampliado para uma definição maior de jogo em geral (HUIZINGA, 1971). Jogo é um fato mais antigo que a cultura, transcendendo as necessidades básicas da vida, e conferindo um sentido à ação (HUIZINGA, 1971). Essa definição de jogos não confere a diferença entre jogos e jogos digitais. Os jogos digitais são sistemas, que possuem o computador como parte de seu sistema, mas que não representa o jogo (SALEN; ZIMMERMAN, 2012a). Neste trabalho, os objetos de estudo são os jogos digitais.

Jogos Digitais, no contexto deste trabalho, significa o domínio de aplicação, para o qual se pretende propor soluções, ou ainda direcionamentos de soluções, para desafios recorrentes. Como esse domínio é complexo, e os desafios são vários, optou-se por focar em um desafio em particular, sendo esse conhecido como *Matchmaking*.

Matchmaking, no contexto de jogos, é o processo de se organizar um conjunto de jogadores em equipes para que possam jogar uns contra os outros (CLAYPOOL M., 2015). Apesar de ser um olhar centrado em desafio único, *Matchmaking* é visto, por alguns autores (NOGUEIRA, 2015), como um dos principais processos que impactam no sucesso de um jogo, além de ser uma tarefa considerada árdua para ser realizada.

NOGUEIRA (2015), em sua dissertação de mestrado, intitulada "Um Estudo sobre a Formação de Equipes em Jogos Virtuais", coloca que: "... formar equipes equilibradas, de forma a ampliar uma disputa não é tarefa fácil, especialmente em um cenário com

milhões de usuários.".

É consenso, entre vários autores e jogadores, que a formação de equipes com níveis de habilidades desbalanceadas entre os jogadores pode impactar negativamente a jogabilidade do jogo e a experiência de seus jogadores. Em uma breve análise, são vistas diversas reclamações, em fóruns de jogos, de jogadores descontentes com partidas desbalanceadas em nível de habilidade entre os mesmos, conforme pode ser visto em um fórum da Riot Games (RIOTGAMES, 2021).

Um dos gêneros de jogos em que os Sistemas de *Matchmaking* estão mais presentes é o *Multiplayer Online Battle Arena* (MOBA). Nos MOBAs, os jogadores controlam um único personagem em um dos dois times participantes, com o objetivo de destruir a base do time inimigo. MOBAs atraem milhões de jogadores, além de uma receita bilionária. O jogo *League of Legends* (LOL) (LEGENDS, 2022), um dos jogos mais populares do gênero, obteve, em 2020, uma receita estimada em 1,75 bilhões de dólares (DACANAY, 2021).

O principal objetivo de um *Matchmaking* efetivo é que os jogadores participantes considerem a partida divertida. Segundo a teoria do *flow* (CSIKSZENTMIHALYI, 1990), para que uma pessoa aproveite uma experiência de forma satisfatória, ela deve entrar em um estado de *flow*. Para que isso aconteça, é necessário que a dificuldade da tarefa realizada seja consistente com o nível de habilidade da pessoa. Caso a tarefa seja muito fácil, ela poderá entrar em um estado de tédio. Já caso seja muito difícil, poderá entrar em um estado de frustração.

Em jogos *online*, a responsabilidade de determinar esse nível de dificuldade de uma partida é do *Matchmaking*. No caso de um MOBA, o jogo deve formar duas equipes de maneira que ambas tenham chances de ganhar a partida. Dessa forma, o jogo não será nem muito fácil, nem muito difícil, para seus participantes (CLAYPOOL M., 2015). No intuito de atingir esse objetivo, dois aspectos são necessários, sendo: (i) estabelecer uma forma de coletar estatísticas dos jogadores, para que existam informações suficientes para determinar o nível de habilidade de um jogador, e (ii) um conjunto de regras para definir a formação das equipes.

Existem ferramentas e estratégias, como o Sistema de *Elos* ou o *Trueskill* (HERBRICH T. MINKA, 2006), que são responsáveis por ranquear o nível de habilidade dos jogadores. Entretanto, o conjunto de regras que deve ser utilizado pelo *Matchmaking* é diferente e inerente para cada jogo.

1.2 Justificativa

Não importa quão bem projetado é um jogo *online*, o pareamento justo entre jogadores sempre vai ser um aspecto fundamental para que a diversão do jogador seja satisfeita. Enfrentar um oponente, com um nível de habilidade muito maior ou muito menor, pode gerar uma sensação de frustração para o jogador, impactando diretamente a sua experiência (VÉRON; MARIN; MONNET, 2014).

Mensurar o nível de habilidade dos jogadores pode ser uma tarefa bem complicada, dependendo do perfil e da complexidade do jogo. Jogos em que jogadores são distribuídos em "papéis", como o LOL (LEGENDS, 2022), têm-se que as qualificações no jogo variam bastante, dependendo de qual papel o jogador irá assumir dentro do jogo (ALMAN; MCKAY, 2017); ou até mesmo com qual personagem o jogador possui maior familiaridade. Ambas, a variedade de estilos de jogos e a demanda por diferentes tipos de habilidade, podem tornar o desenvolvimento da estratégia de *Matchmaking* do jogo mais complexa.

Estratégias estatísticas, como o *Elo Rating System*, podem não ser suficientes para trazer a qualificação real de um jogador. O *Elo Rating System* é uma estratégia que foi construída no Xadrez (ELO, 1978). Para Arpad Emmerich Elo, em ELO (1978), "Um sistema de classificação adequado deve ir um passo além da mera classificação e deve fornecer alguma estimativa dos pontos fortes relativos dos concorrentes.". Essa colocação de Arpad Emmerich Elo atende muito bem aos complexos jogos *online* de hoje, tais como: o *League of Legends* (LEGENDS, 2022) e o *Dota 2* (DOTA2, 2022). Entretanto, o Xadrez possui habilidades que podem ser generalizadas para qualquer partida dentro do seu jogo, o que é diferente de jogos, como o *League of Legends* (LEGENDS, 2022), ou outros jogos *online*.

Pensando em uma análise mais específica para a classificação de habilidade dos jogadores, percebe-se uma possibilidade de uso de cognição, responsável por realizar uma metrificação mais aprimorada sobre o nível de habilidade de um jogador. Entende-se por cognição, de acordo com SIGNIFICADOS (2022), como sendo a "capacidade de processar informações e transformá-las em conhecimento, com base em um conjunto de habilidades mentais e/ou cerebrais como a percepção, a atenção, a associação, a imaginação, o juízo, o raciocínio e a memória.".

Nesse contexto, há um modelo conceitual pertinente, e que propõe o uso de entidades inteligentes - Agentes de Software, para lidar com esse processo orientado à cognição. Um Agente de Software possui recursos importantes, tais como: realizar ações autônomas e interagir com outros Agentes (WOOLDRIDGE, 2009). No momento em que os Agentes de Software começam a interagir entre si, trocando mensagens por meio de alguma estrutura de comunicação computacional, eles formam um Sistema Multiagentes (SMA) (WOOLDRIDGE, 2009).

Os Sistemas Multiagentes possuem potencial, sendo utilizados para lidarem com uma série de problemas na Indústria (BELLIFEMINE; CAIRE; GREENWOOD, 2007). Suas capacidades de comunicação e autonomia podem ser utilizadas para processamento e filtragem de informações (BELLIFEMINE; CAIRE; GREENWOOD, 2007) (KLUSCH, 2001). Diante do exposto, pressupõe-se que o uso dessa capacidade de processamento de informações pode tornar os SMAs interessantes para projetar um sistema de estratégia de *Matchmaking*, no domínio de jogos. Essa é uma premissa no escopo deste trabalho.

Ainda justificando a premissa, a capacidade dos SMA em gerenciar alto processamento de informações pode ser um fator de relevância, bastante pertinente, para processar informações de jogadores dentro de um fluxo de pedidos para participar de uma partida. Essas informações podem ser coletadas por Agentes de Software, sendo repassadas para outros agentes, que podem atuar como entidades colaborativas. Essa atuação colaborativa dos Agentes de Software, combinada às estratégias citadas anteriormente (i.e *Elo Rating System* e *Trueskill*), pode auxiliar na filtragem de informações quanto às habilidades dos jogadores que são mais relevantes para o jogo. Consequentemente, pode-se definir partidas entre jogadores de uma forma mais justa e uniforme, não havendo uma disparidade de habilidades entre eles.

1.3 Questão de Pesquisa

Perante o que foi apresentado na Contextualização e na Justificativa, a questão de pesquisa é: Sistemas Multiagentes são aderentes para o desenvolvimento de Sistemas *Matchmaking* de Jogos *Online*? A pesquisa foca em autores reconhecidos da área de *Matchmaking*, tais como: ELO (1978), HERBRICH T. MINKA (2006), GLICKMAN (1995); bem como em Sistemas Multiagentes, tais como: BELLIFEMINE; CAIRE; GREENWOOD (2007) e WOOLDRIDGE (2009).

Cabe colocar que essa questão de pesquisa parte da hipótese de que Sistemas Multiagentes são pertinentes para tratar Sistemas *Matchmaking* de Jogos *Online*.

Ao final do trabalho, portanto, é revelado se essa hipótese é de fato satisfeita; parcialmente satisfeita, ou ainda refutada. Se satisfeita ou parcialmente satisfeita, então a questão de pesquisa será respondida apresentando os resultados positivos do uso de Sistemas Multiagentes em Sistemas *Matchmaking* de Jogos *Online*. Caso contrário, a questão de pesquisa será respondida apresentando as preocupações observadas com o uso de Sistemas Multiagentes em Sistemas *Matchmaking* de Jogos *Online*.

Por fim, pretende-se também validar ou refutar a hipótese de que a solução, se satisfatória, também é adaptável a qualquer jogo.

1.4 Objetivos

1.4.1 Objetivos Gerais

O objetivo geral da pesquisa é saber se Sistemas Multiagentes são ou não pertinentes para Sistemas *Matchmaking*. Para viabilizar esse objetivo, ocorreu a necessidade de implementar Multiagentes em um contexto *Matchmaking*. Portanto, optou-se pelo desenvolvimento de uma API orientada a Sistemas Multiagentes, conferindo cognição ao *Matchmaking*, e sendo adaptável o suficiente para que os desenvolvedores de jogos possam utilizá-la em seus jogos.

1.4.2 Objetivos Específicos

Dentre os Objetivos Específicos, estão:

- Investigar Sistemas Multiagentes, em termos de literatura especializada;
- Investigar Sistemas *Matchmaking*, em termos de literatura especializada;
- Especificar, Projetar, Desenvolver e Documentar uma API, orientada a SMA, capaz de auxiliar na criação de algoritmos de *Matchmaking*, e
- Analisar os Resultados Obtidos, procurando documentar as impressões percebidas, com foco na questão de pesquisa.

1.5 Organização da Monografia

Este trabalho está organizado conforme os seguintes capítulos:

- **Capítulo 2 - Referencial Teórico:** introduz o domínio, no qual o trabalho está inserido, no caso, Domínio de Jogos *Online*; conceitos, princípios e práticas a respeito do Paradigma de Sistemas Multiagentes, e ainda algoritmos e ferramentas de *Matchmaking* já disponíveis no mercado, conferindo uma breve apresentação de trabalhos relacionados;
- **Capítulo 3 - Suporte Tecnológico:** apresenta e explica as ferramentas e outros apoios tecnológicos utilizados na realização desse trabalho;
- **Capítulo 4 - Metodologia:** explica os detalhamentos metodológicos estabelecidos para a confecção desse trabalho, com foco em pesquisa, desenvolvimento e análise de resultados;

- **Capítulo 5 - *Multiagents Matchmaking API*:**apresenta as especificações da API, chamada *Multiagents Matchmaking API*, considerando a arquitetura de comunicação do Sistema Multiagentes, os requisitos da API, bem como uma breve explicação de como ocorreu o desenvolvimento da mesma, considerando a metodologia de desenvolvimento estabelecida;

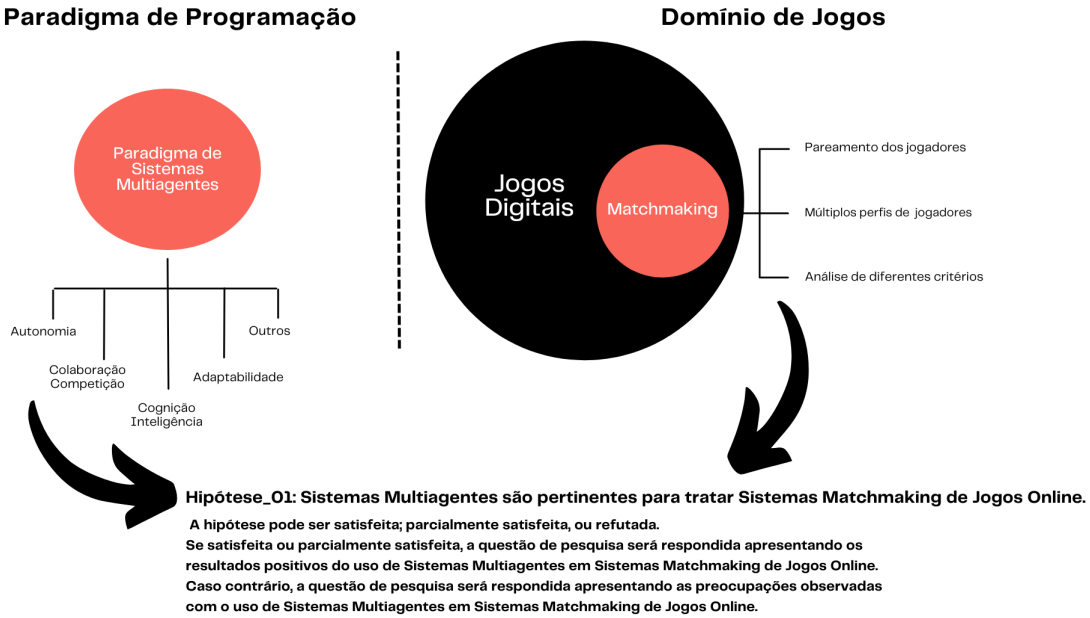
- **Capítulo 6 - Análise de Resultados:** apresentação detalhada dos resultados obtidos, considerando a metodologia de análise de resultados estabelecida, centrada em Pesquisa-ação, e

- **Capítulo 7 - Conclusão:** confere um fechamento ao projeto, retomando seu propósito, seus objetivos, e procurando responder a questão de pesquisa. Por fim, são acordadas ideias para trabalhos futuros.

2 Referencial Teórico

Conforme colocado na Introdução, o presente trabalho tem como domínio de interesse os Jogos Digitais. Adicionalmente, pretende-se contribuir para esse domínio orientando-se por um paradigma de programação, chamado Sistema Multiagentes. Essa contribuição parte de hipóteses, dentre elas: Sistemas Multiagentes são pertinentes para tratar Sistemas *Matchmaking* de Jogos *Online*, uma vez que os agentes de software orientam-se por um modelo conceitual apoiado em autonomia, colaboração/competição, adaptabilidade e outras características. Portanto, cabe ainda conceituar *Matchmaking*. A Figura 1 ilustra os principais elementos envolvidos na fundamentação desse trabalho.

Figura 1 – Representação do Domínio do Trabalho



Fonte: autores

Neste capítulo, será apresentado o embasamento conceitual do trabalho, com olhar para: o Domínio de Jogos Digitais; o Paradigma de Sistema Multiagentes, e o Sistemas *Matchmaking*. Por fim, tem-se o Resumo do Capítulo.

2.1 Domínio de Jogos Digitais

Segundo SALEN; ZIMMERMAN (2012a), "os jogos digitais e eletrônicos assumem uma grande quantidade de formas, e são projetados para muitas plataformas e computadores diferentes.". Os mesmos autores ainda trazem a definição de que jogos digitais são sistemas, e que precisam de um meio, um computador ou um console, para serem utili-

zados por seus jogadores. Esses jogos digitais podem ser projetados para serem jogados por um único jogador ou por vários, existindo uma interação entre eles. Todos esses jogos possuem expectativas por parte de seus jogadores, que esperam a melhor experiência possível, aguardando alto nível de *gameplay*, emoção e convencimento por parte do jogo (CHANDLER, 2020).

Muitas pessoas são envolvidas no processo de desenvolvimento de um jogo, não apenas sendo Desenvolvedores de Software, *Designers* de jogos ou Artistas, mas também sendo Investidores, Testadores de Qualidade e muitos outros (CHANDLER, 2020). Essa cadeia de desenvolvimento de um jogo digital acaba se tornando complexa, envolvendo uma grande quantidade de pessoas. Portanto, exige uma responsabilidade alta de comunicação entre as partes.

Apesar das diversidades de desenvolvimento, a base de jogadores ativos e a quantidade de jogos em circulação no mercado têm crescido cada vez mais. Durante o ano de 2020, início da pandemia da Covid-19, o mercado de jogos digitais cresceu em cerca de 12%, arrecadando uma receita total de 127 bilhões de dólares (VALENTINE, 2021). Os jogos que são gratuitos para jogar (*free-to-play*), que possui uma forte base de multijogadores, obtiveram a maior parte dessa receita, aproximadamente 78% (VALENTINE, 2021). O Brasil, em 2021, tornou-se o 12º maior fabricante de jogos no mundo, com um crescimento de 2,3 bilhões de dólares (ABRAGAMES, 2021). Tais evidências mostram o quanto jogos digitais estão em crescimento no mundo, e como o seu mercado vem demandando maior qualidade sobre o seu desenvolvimento.

A expansão do mercado de jogos digitais tem impactado diretamente na rotina das pessoas. Não é novidade que esses jogos estão entrando cada vez mais na vida das pessoas, como por exemplo na educação. Os vídeo games estão proporcionando diferentes tipos de ambientes, sendo esses focados em aprendizagem, e mais eficazes em motivar e engajar os seus alunos para conseguirem resolver problemas (OREN; PEDERSEN; BUTLER-PURRY, 2021). Os autores OREN; PEDERSEN; BUTLER-PURRY (2021) trazem, em seu artigo, que revisões da literatura sugerem que os vídeo games possuem um impacto mais positivo na educação do estudante do que outros tipos de mídia. Isso ocorre, uma vez que os alunos estão envolvidos com uma situação diferente da qual estariam no cotidiano. Sendo assim, os jogos digitais buscam aprimorar e provocar estímulos nesses estudantes, os quais não seriam incentivados ao usar metodologias mais convencionais de ensino.

O impacto dos jogos digitais na educação pode ser visto no ensino de estudantes de Engenharia de Software. O ensino da Engenharia de Software vai muito além da programação, estendendo-se à gestão, à qualidade, entre outras áreas complementares. O uso de Aprendizado Baseado no Desenvolvimento de Jogos (GDBL) pode se tornar útil, dado que os jogos despertam o interesse dos alunos, sendo atraentes para explicar, por exemplo, o processo de desenvolvimento de um software (SOUZA et al., 2017).

2.1.1 Desenvolvimento de Jogos Digitais

Como apresentado anteriormente, o desenvolvimento de jogos digitais tem se tornado cada vez mais complexo. Tomando como fato o aumento da base de jogadores, a cobrança pelo aperfeiçoamento desses jogos também aumenta. Construir um jogo, hoje, compreende uma experiência diferente do que, por exemplo, em 1994, onde os jogos, em sua maioria, utilizavam gráficos 2D, não incorrendo na necessidade de se preocuparem com uma terceira dimensão, mas demandando cuidado no uso de recursos, uma vez que o *hardware* impunha maior limitação, e as plataformas de desenvolvimento não promoviam facilitadores vistos hoje (BLOW, 2004).

O desenvolvimento de um jogo digital tornou-se muito rico à medida que o tempo passou. Por consequência, são projetos de software de larga escala. Ferramentas, *frameworks*, bibliotecas, motores, linguagens de programação, entre outros, também acompanharam essa tendência, por um lado, facilitando o trabalho dos desenvolvedores; por outro, adicionando complexidade aos jogos.

Diferentes metodologias de desenvolvimento de projetos de software podem ser utilizadas para o desenvolvimento de um jogo, das quais as mais conhecidas acabam sendo cascata, espirais e ágeis (FATIMA; RASOOL; QAMAR, 2018). Existem pesquisadores que estudam o desenvolvimento de jogos, e buscam estudar as práticas da Engenharia de Software para tentar aprimorar esse desenvolvimento de jogos (FATIMA; RASOOL; QAMAR, 2018).

Ainda explorando as metodologias de desenvolvimento de jogos como software, o uso das iterações é bastante estimulado nesse processo, tornando-se uma parte relevante (KULTIMA, 2015). KULTIMA (2015) menciona que foram feitas pesquisas em empresas de jogos Finlandesas, entre 2013 e 2014. Nessas pesquisas, de 105 menções à palavra "iteração", feitas pelos entrevistados, cerca de 77 menções foram usadas no intuito de colocar que o desenvolvimento de jogos é iterativo por natureza. Essa concepção mostra que, para os profissionais dessas empresas de desenvolvimento de jogos, a solução na criação desses jogos sempre precisa de refinamentos constantes, buscando oferecer a melhor experiência de jogo para o usuário final (KULTIMA, 2015).

Oferecer a melhor experiência de usuário em um jogo é fundamental para o seu público. A experiência de usuário, na grande maioria dos jogos, compreende a parte principal do seu desenvolvimento. Entretanto, essa é uma parte difícil de se desenvolver. A experiência de um jogo é uma interação lúdica, e que o *designer* do jogo não a cria de maneira direta (SALEN; ZIMMERMAN, 2012b). Como diz SALEN; ZIMMERMAN (2012a), "O *designer* de jogos apenas projeta indiretamente a experiência do jogador, criando diretamente as regras". O *designer* não pode, necessariamente, estimar como cada jogador irá vivenciar aquele jogo. Entretanto, ele pode estipular regras no jogo, nas quais os jogadores

poderão explorá-las e vivenciá-las.

2.2 Paradigma de Sistemas Multiagentes

Para explorar o Paradigma de Sistema Multiagentes, é necessário, em um primeiro momento, entender algumas particularidades desse modelo conceitual, tais como: O que é um Agente?, e Como Funciona um Sistema Multiagentes?

2.2.1 O que é um Agente?

Existem diversas definições para agentes. Entretanto, de acordo com MCARTHUR et al. (2007), mesmo que as definições sejam diferentes, elas compartilham um conjunto de conceitos, como a noção de um agente, seu meio e sua autonomia. Conforme colocado por WOOLDRIDGE (2009), um agente é uma entidade que pertence a um meio, sendo capaz de reagir de forma autônoma às mudanças do meio.

O meio, ou o ambiente, é tudo aquilo externo ao agente. Para que o agente esteja situado no meio, é necessário que uma parte do ambiente possa ser observada ou alterada pelo agente (WOOLDRIDGE, 2009).

Existem seis características ortogonais que podem ser usadas para caracterizar um agente (GRISS; HEINEMAN; COUNCILL, 2001), sendo elas:

- **Adaptabilidade**: representa o quanto um agente pode mudar seu comportamento após a implementação;
- **Autonomia**: representa o nível em que o agente é responsável pelo seu próprio controle e pode buscar seus próprios objetivos;
- **Colaboração**: representa o quanto um agente consegue se comunicar e cooperar com outros agentes;
- **Inteligência**: representa a capacidade do agente em raciocinar a respeito de suas metas e conhecimentos;
- **Mobilidade**: representa a capacidade do agente de mudar de contexto, seja começando no novo contexto do zero, ou mantendo seu estado de um contexto para o outro, e
- **Persistência**: representa a capacidade do agente de armazenar conhecimento e estado no decorrer do tempo, e no caso de erros.

Além de cada agente possuir características distintas, eles podem se orientar por uma das duas categorias seguintes:

- **Agentes comportamentais**: um tipo de agente reativo, sendo suas ações baseadas em comportamentos do mundo real, como comer, andar, correr, entre outros (BROOKS, 1991), e
- **Agentes intencionais**: um tipo de agente capaz de decidir sozinho seus objetivos, e os mudar no decorrer do tempo, abstraindo conceitos como crenças, desejos e intenções para melhorar seu raciocínio e sua capacidade de aprendizado (BRATMAN, 1999).

2.2.2 Como funciona um Sistema Multiagentes?

Um Sistema Multiagentes nada mais é do que um sistema que possui pelo menos dois agentes trabalhando em busca de um objetivo (MCARTHUR et al., 2007). É importante notar que os objetivos a serem alcançados referem-se aos objetivos de cada agente, não um objetivo comum do sistema (WOOLDRIDGE, 2009).

A arquitetura de Sistemas Multiagentes é o mecanismo fundamental para a estruturação dos agentes autônomos, em busca de uma representação do mundo real. Tais arquiteturas dividem-se entre modelos totalmente lógicos, baseados nos agentes comportamentais; até modelos totalmente orientados a metas, baseados em agentes intencionais. Além desses dois modelos, existem modelos híbridos, os quais se orientam por ambos os tipos de agente (BELLIFEMINE; CAIRE; GREENWOOD, 2007).

2.3 *Matchmaking*

A palavra *Matchmaking*, segundo DICTIONARY (2022), "é a atividade de arranjar casamentos ou relacionamentos românticos entre pessoas". Essa definição mostra que, no surgimento da palavra, a ideia de *Matchmakings* era correlacionada, principalmente a arranjos de casamentos. Na Inglaterra por exemplo, em 1600, sugiram as primeiras agências de encontros, com sacerdotes da igreja responsáveis por arranjar aos seus fiéis um cônjuge de mesma classe social (WALKER, 2022). Hoje em dia, a palavra *Matchmaking* é muito referida em aplicativos e *sites* de romances, responsáveis por buscar encontros entre pessoas no mundo inteiro.

Apesar da palavra *Matchmaking* ainda estar muito relacionada a encontros românticos, ela também ganhou muita força no mundo dos jogos, principalmente no Xadrez. Em vários esportes e jogos sempre foram levantados muitos questionamentos sobre qual jogador era melhor que o outro (ELO, 1978). Esses questionamentos dão-se por base, para poder criar sistemas de classificação de jogadores. Para ELO (1978), sistemas de classificação são responsáveis por comparar jogadores em pares ou equipes individuais. Essa

necessidade de comparação surge, devido à necessidade de montar partidas que fossem mais equilibradas entre os seus oponentes.

Atualmente, os Sistemas de *Matchmaking* são muito explorados pelos jogos *onlines*. Quando um jogador deseja jogar um jogo *multiplayer online*, como o *League of Legends* (LEGENDS, 2022) ou o *Dota 2* (DOTA2, 2022), ele passa a ocupar uma posição, em uma fila de jogadores, aguardando o servidor do *Matchmaking* parear os jogadores entre si (MINKA; CLEVEN; ZAYKOV, 2018). O servidor do *Matchmaking* é responsável por analisar diversos critérios, e tomar uma decisão para o pareamento dos jogadores. Esses critérios são decididos pelos projetistas do jogo, que usam algoritmos, como o *"Elo Rating System"*, para montar partidas mais balanceadas e justas para os seus jogadores.

No entanto, a projeção desses sistemas de *Matchmaking* de jogos não é simples. À medida que os jogos vão adquirindo maiores níveis de complexidade, como uma nova habilidade dentro do jogo, uma nova armadilha, uma alteração no mapa principal do jogo, entre outras diversas mudanças, isso causa um impacto direto sobre o como o *Matchmaking* deve olhar no momento em que vai gerar uma nova partida. Outro fator que causa forte influência é uma partida em equipe. Por mais que níveis de habilidade possam ser metrificados, conforme acordado de forma didática em PEREIRA; ROQUE (2012), é muito difícil coletar a sinergia da equipe, para poder determinar se ela é uma equipe equilibrada para a partida ou não (DENG et al., 2021). Nesse cenário, um fator que precisa ser considerado, e que influencia muito no *Matchmaking*, é a Experiência do Jogador

2.3.1 Experiência do Jogador

Os *Matchmakings* impactam diretamente na experiência dos jogadores em jogos *online*. VÉRON; MARIN; MONNET (2014) consideram três fatores principais para a experiência do jogador:

- **Tempo de espera do jogador dentro da fila do *Matchmaking* até a formação de uma partida**: A demora para a formação de uma partida para jogadores tem um impacto sobre a experiência. Isso é um problema, principalmente, para jogadores que são mais habilidosos. Devido à escassez de outros jogadores do mesmo nível, cria-se uma dificuldade para o servidor do *Matchmaking* combinar os jogadores entre si para uma partida;

- **Precisão da formação da partida entre os jogadores**: Combinar os jogadores com o mesmo nível de habilidade é fundamental. Quando esse fator não é considerado, jogadores que são mais inexperientes podem se sentir frustrados contra oponentes mais fortes, e o mesmo pode ocorrer para os mais habilidosos, gerando uma sensação de falta de desafios e recompensas, e

- **Latência do tempo de resposta do servidor entre os jogadores**: Além dos dois últimos fatores, a combinação dos jogadores deve levar em consideração o tempo de latência que eles possuem com o servidor. Partidas onde o *ping* é alto, por exemplo, costumam impactar negativa e sensivelmente a jogabilidade.

Cada um desses problemas é trabalhado de forma diferente por diversos autores. XU; YU; WU (2021) buscam compreender o "**Tempo de espera do jogador dentro da fila do *Matchmaking* até a formação de uma partida**". No trabalho proposto, os autores estudam o tempo de espera por meio de uma estrutura de filas, e buscam compreender o impacto de diferentes regras estáticas e dinâmicas de *match* nesse tempo de espera.

DENG et al. (2021) abordam a "**Precisão da formação da partida entre os jogadores**". É proposto um modelo de otimização global de *Matchmaking*, denominado *GloMatch*, utilizando aprendizado de máquina. Esses autores transformaram o problema de *Matchmaking* em um problema de decisão sequencial, e formalizaram o seu processo em um Processo de Decisão de Markov (PELLEGRINI; WAINER, 2007). A ideia é pesquisar combinações adequadas de jogadores, e melhorar a satisfação do jogador com o sistema de *Matchmaking*, criando uma sensação de que as suas partidas são mais justas e precisas.

Já AGARWAL; LORCH (2009) possuem um olhar mais criterioso em relação à "**Latência do tempo de resposta do servidor entre os jogadores**". Foi implementado um sistema denominado *Htrae*, que é um sistema de previsão de latência, responsável por prevenir sobre a combinação de jogadores que possuem uma alta latência entre si no jogo. Os autores explicam que o *Htrae* utiliza um sistema de geolocalização, sendo esse combinado a um Sistema de Coordenada de Rede, responsável por atribuir, ao dispositivo em que o jogador está utilizando, uma coordenada em um espaço métrico virtual, de modo que a distância entre dois dispositivos seja equivalente ao tempo de ida e volta de uma resposta entre elas.

A dificuldade para a construção do *Matchmaking* "ideal"deve-se, principalmente, aos três problemas apontados anteriormente. A forma como esses problemas podem ser mitigados ou amenizados, para não prejudicar a experiência do jogador, depende muito de vários fatores, desde à complexidade do jogo, ao comportamento da comunidade que joga o jogo estudado.

2.3.2 Algoritmos de *Matchmaking*

São vários os algoritmos que se propõem a montar partidas entre jogadores. Cada um desses algoritmos possuem as suas devidas particularidades, sendo pensados para cenários específicos, seja o *Elo Rating System* (ELO, 1978) planejado para o Xadrez; o *Glicko* (GLICKMAN, 1995) em resposta a deficiências que existem no sistema *Elo*

(ELO, 1978), ou o *TrueSkill* (HERBRICH T. MINKA, 2006) desenvolvido pela Microsoft (MICROSOFT, 2022) para os seus jogos, tais como: o *Halo* (HALO, 2022) e o *Gears of War* (WAR, 2022). Nas próximas seções, seguem outros detalhamentos sobre cada um desses algoritmos.

2.3.2.1 *Elo Rating System*

O *Elo Rating System* foi criado por um professor de física e mestre em xadrez, chamado Arpad Elo (ELO, 1978). Elo decidiu a lógica dos sistemas de classificação de sua época, com o objetivo de entender as suas fraquezas e desenvolver um sistema que fosse baseado na teoria estatística e na probabilidade (ELO, 1978). O *Elo Rating System* é uma abordagem científica para a avaliação de desempenho do Xadrez, e que, posteriormente, foi utilizada para outros jogos. O modelo é um sistema numérico, no qual a diferença de avaliação dos jogadores pode ser convertida em probabilidade de vitória (ELO, 1978). O *Elo Rating System* pode ser formalizado da seguinte forma:

Seja Ea a estimativa de vitória do jogador A, representada pela Equação 2.1, e Eb a estimativa de vitória do jogador B, representada pela Equação 2.2.

$$Ea = \frac{1}{1 + 10^{\frac{Rb - Ra}{400}}} \tag{2.1}$$

$$Eb = \frac{1}{1 + 10^{\frac{Ra - Rb}{400}}} \tag{2.2}$$

Onde Ra e Rb são as classificações específicas do jogador A e do jogador B.

Para poder atualizar a classificação dos jogadores, após uma partida, é apresentada a seguinte fórmula:

Seja Ra' a nova classificação do jogador A, representada pela Equação 2.3, e Rb' a nova classificação do jogador B, representada pela Equação 2.4.

$$Ra' = Ra + K(Sa - Ea) \tag{2.3}$$

$$Rb' = Rb + K(Sb - Eb) \tag{2.4}$$

Onde, Sa e Sb podem ser respectivamente 0 ou 1 para caso o jogador tenha perdido ou vencido a partida, respectivamente. Na fórmula, é possível ver K, sendo essa uma constante de ajuste das classificações. Caso K seja muito grande, a classificação dos jogadores irá sofrer um impacto igualmente grande.

2.3.2.2 *Glicko*

O Sistema de Elo (ELO, 1978) foi utilizado em diversos torneios de Xadrez, tornando-se bastante popular no mundo inteiro. Entretanto, apesar de ter sido uma grande evolução em relação a outros modelos em seu tempo, ele apresentava alguns defeitos. Diante disso, surge o *Glicko* (GLICKMAN, 1995). GLICKMAN (1995) tenta, com o *Glicko*, resolver o problema de confiabilidade da classificação do jogador no Sistema de Elo (ELO, 1978). O *Glicko* estende o *Elo Rating System* (ELO, 1978), calculando não só a estimativa de força no jogo, mas também adicionando um desvio da classificação, responsável por medir a incerteza em uma classificação (GLICKMAN, 1995). O desvio, quando está alto, indica que um jogador competiu poucos jogos de torneio. Já o desvio baixo indica que o jogador compete com frequência. O cálculo do *Glicko* pode ser formalizado como consta a seguir.

Todo o seu cálculo é baseado em estimativas anteriores do jogador. Caso o jogador não possua um histórico de partidas, a sua classificação é definida no valor de 1500 com um desvio (RD) de 350. Para o caso do jogador já possuir um histórico de partidas, seja o RD' seu desvio de classificação antigo e RD seu novo desvio, ele pode ser determinado pela Equação 2.5.

$$RD = \min(\sqrt{RD'^2 + c^2}, 350) \tag{2.5}$$

Reparando em c, ela é uma constante responsável por gerir o grau de incerteza entre os desvios das classificações.

No *Glicko*, a classificação do jogador muda apenas levando em consideração o resultado do seu jogo. Entretanto, o desvio leva em consideração não só o resultado do jogo, como também a frequência de jogos em que o jogador tem participado.

2.3.2.3 *TrueSkill* 1 e 2

O *TrueSkill*, segundo HERBRICH T. MINKA (2006), é um Sistema Bayesiano de Classificação de Habilidades, sendo visto como uma generalização do Sistema de Elo (ELO, 1978). É fato que o Sistema de Elo (ELO, 1978) revolucionou a área de jogos, sendo fundamental para os estudos sobre *Matchmaking*, no domínio de jogos. Portanto, assim como no Glicko (GLICKMAN, 1995), o *Elo Rating System* (ELO, 1978) também teve influência sobre o *TrueSkill* (HERBRICH T. MINKA, 2006).

No *TrueSkill*, os dados são extraídos de uma sequência de resultados de jogos, onde são listados os jogadores envolvidos na partida, atribuições desses jogadores, duração da partida, tempo de jogo e pontuação final (MINKA; CLEVEN; ZAYKOV, 2018). O *Trueskill* realiza as seguintes operações:

- A habilidade (*skill*) inicial do jogador, em um tempo t. A habilidade é extraída de uma distribuição gaussiana com uma média m0 e variância v0, sendo determinada pela Equação 2.6;
- Após cada partida, a habilidade (skill) do jogador é recalculada seguindo a Equação 2.7. Sendo L, a duração da partida;
- O próximo passo é calcular o desempenho, em inglês *performance* (perf), do jogador a partir da sua habilidade. O cálculo do desempenho é feito através da Equação 2.8. Sendo β um parâmetro ajustável, que representa a quantidade de aleatoriedade no jogo, e
- Em um jogo com times, o desempenho de cada time é definido pela Equação 2.9.

$$skill_i^{t_0} \sim \mathcal{N}(m_0, v_0) \tag{2.6}$$

$$skill_i^{t+L} \sim \mathcal{N}(skill_i^t, \gamma^2) \tag{2.7}$$

$$perf_i^t \sim \mathcal{N}(skill_i^t, \beta^2) \tag{2.8}$$

$$perf_{team} = \sum_{i \epsilon team} perf_i \frac{timePlayed_i}{L} \tag{2.9}$$

O resultado esperado pelo *Trueskill* é determinado pelo desempenho dos dois times (HERBRICH T. MINKA, 2006).

No *TrueSkill*, o grau de incerteza sobre a habilidade do jogador determina se o sistema conhece ou não o nível de habilidade do jogador. Sendo assim, se a incerteza for grande, o nível de habilidade ainda é incerto. Caso contrário, o sistema acredita que o nível de habilidade do jogador está próximo da habilidade média do jogo (HERBRICH T. MINKA, 2006).

O *TrueSkill* 2 é uma evolução do *TrueSkill*. A ideia do segundo sistema é que não seja apenas levada em consideração a quantidade de partidas vitoriosas, ou seja, uma análise mais generalista sobre o jogador. Adicionalmente, com o *TrueSkill* 2, a análise deve considerar as habilidades individuais do jogador (MINKA; CLEVEN; ZAYKOV, 2018), tais como: tempo de partida, quantidade de pontos efetuados na partida e entre outros.

2.4 Resumo do Capítulo

Neste Capítulo, foi introduzido o domínio de jogos digitais, contando uma breve história de como se deu a evolução até os jogos atuais, além de contextualizar o processo de desenvolvimento de um jogo, apresentando algumas metodologias associadas.

Em um segundo momento, foram abordados os conceitos relacionados aos Sistemas Multiagentes. Nessa seção, foram apresentadas: a definição de um agente de software; as características dessa entidade inteligente, bem como as diferentes vertentes que orientam a implementação de um agente (i.e. comportamental, intencional, ou híbrida). Na sequência, foi explicado como um Sistema Multiagentes funciona.

Por fim, detalhou-se sobre *Matchmaking*, com foco em: onde surgiu, o que significa, e como se enquadra no contexto dos jogos digitais. Essa seção do capítulo é muito importante para que se entenda o impacto do *Matchmaking* na experiência dos jogadores. Ainda nessa seção, são descritos alguns algoritmos, associados aos principais sistemas disponíveis no mercado, e dedicados à realização de *Matchmakings* em jogos.

3 Suporte Tecnológico

Este capítulo apresenta as principais ferramentas e tecnologias utilizadas durante o desenvolvimento desse trabalho, justificando cada escolha. O suporte tecnológico confere tecnologias utilizadas: (i) no Apoio à Pesquisa, facilitando o processo de levantamento bibliográfico; a organização de referenciais, e a escrita da monografia, e (ii) no Apoio ao Desenvolvimento, acordando linguagens, *frameworks* e ferramentas. Por fim, tem-se o Resumo do Capítulo.

3.1 Tecnologias de Apoio à Pesquisa

3.1.1 *LaTeX*

O *LaTeX* é um sistema de preparação de documentos, utilizado com frequência para produção de documentos técnicos ou científicos (LATEX, 2022). O *LaTeX* é utilizado, nesse trabalho, para auxiliar na formatação da monografia, permitindo aos autores concentrarem seus esforços no conteúdo a ser escrito.

3.1.2 *Overleaf*

Overleaf é uma ferramenta colaborativa *online* para escrita de documentos científicos (OVERLEAF, 2022). O *Overleaf* é utilizado, nesse trabalho, por conta da facilidade que provê na escrita e na edição do documento, além de permitir que ambos os autores consigam elaborar colaborativa e simultaneamente a monografia de forma *online*.

3.1.3 *Mendeley*

Mendeley é uma ferramenta de gestão de referências (MENDELEY, 2022). Está sendo utilizada, nesse trabalho, por prover a possibilidade de compartilhamento de referências entre os autores, além de permitir o uso de comentários nos documentos pesquisados, e a organização dos mesmos, conforme desejado.

3.1.4 *Kaggle*

O *Kaggle* é uma comunidade de online de cientistas de dados e profissionais de aprendizado de máquina (KAGGLE, 2022). Na plataforma do *Kaggle* é possível buscar e acessar por bases de dados abertas que foram usadas nos testes da *Multiagents Matchmaking API*.

3.2 Tecnologias de Apoio ao Desenvolvimento

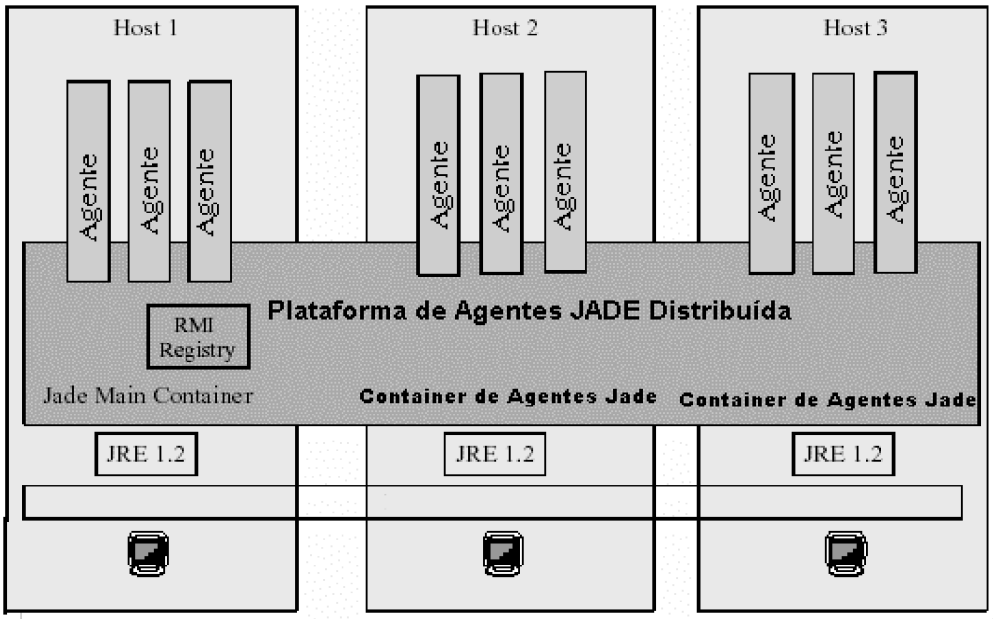
3.2.1 JADE

O JADE é um *framework* implementado na linguagem Java, que simplifica a implementação de Sistemas Multiagentes, além de prover um conjunto de ferramentas gráficas. Essas ferramentas auxiliam nos processos de depuração de erros, e de implantação dos Sistemas Multiagentes (JADE, 2022). O JADE é utilizado, nesse trabalho, para facilitar a criação e a comunicação dos agentes utilizados no sistema de *Matchmaking* proposto.

Esse *framework* coordena as interações entre muitos agentes, orientando-se por protocolos bem definidos, padronizados e estabelecidos pela FIPA (FIPA, 2002a). Dentre as vantagens, cabe mencionar a comunicação facilitada, e a possibilidade de compartilhar e/ou integrar diferentes sistemas, desde que desenvolvidos com esses mesmos protocolos. A plataforma de agentes pode estar distribuída em diferentes máquinas, sem ser necessário o mesmo Sistema Operacional. As configurações podem ser controladas via uma Interface Gráfica remota, sendo possível alterar essas configurações em tempo de execução, e até mesmo mover agentes de uma máquina para outra, quando necessário. Outro ponto interessante, e que foi considerado para escolha dessa ferramenta, compreende o fato da mesma ser gratuita, com *copyright opensource* LGPL, desde Maio de 2003.

Esse *framework* revela uma arquitetura específica, conforme consta na Figura 2.

Figura 2 – Arquitetura da Plataforma JADE



Fonte: JADE (2022)

A Arquitetura do JADE é constituída por *containers* que podem estar distribuídos pela rede. Os *containers* são habitados por agentes. O conjunto desses *containers* é cha-

mado de plataforma, e é responsável por abstrair a complexidade do *hardware*, Sistema Operacional, tipos de Rede, JVM (Java *Virtual Machine*), dentre outros (JADE, 2022).

Um desses *containers* é considerado o principal, o qual representa o ponto de inicialização da plataforma. Ele é responsável por: (i) administrar as referências de objeto e endereço de transporte de todos os *containers*; (ii) gerenciar o registrador de todos os agentes presentes na plataforma, e (iii) hospedar os agentes especiais especificados pela FIPA (FIPA, 2002a).

Esses agentes especiais são:

- **AMS**: agente responsável por supervisionar a plataforma inteira. Trata-se de um ponto de contato para os agentes que precisarem interagir com páginas brancas ou gerenciar seus ciclos de vida. Todo agente precisa estar registrado no Sistema de Gerenciamento de Agentes (em inglês, chamado *Agent Management System* (AMS), controlado por um agente de mesmo nome). Esse sistema é conhecido como páginas brancas;
- **DF**: esse agente implementa o serviço de páginas amarelas, onde os agentes registram os seus serviços, e
- **ACC**: esse agente provê o caminho para que agentes possuam um contato básico dentro e fora da plataforma. A plataforma confere um sistema de transporte, o qual é responsável por prover canais de comunicação de agentes (em inglês, *Agent Communication Channel* (ACC), também controlado por um agente de mesmo nome).

Para implementar em Jade, devem ser atendidos alguns pré-requisitos, dentre eles:

- Máquina virtual java 1.4 ou mais recente;
- Kit de desenvolvimento JDK 1.4 ou mais recente;
- Pasta bin do JDK incluida no PATH do Sistema Operacional;
- Editor de texto ou IDE para desenvolvimento (Notepad, Gedit, Eclipse, Netbeans, Intellij IDEA), e
- Download do JADE (cadastro gratuito) em <http://jade.tilab.com> (último acesso em Setembro de 2022).

3.2.2 RabbitMQ

RabbitMQ é um sistema intermediário de mensagens *open source*, que centraliza o envio de mensagens da aplicação em um lugar seguro (RABBITMQ, 2022). O *RabbitMQ*

é usado, nesse trabalho, como uma fila de mensagens que armazena quais jogadores pretendem jogar, na ordem em que pediram para participar de um jogo.

3.2.3 MongoDB

MongoDB é um banco de dados baseado em documentos, de fácil aplicação e escalável (MONGODB, 2022). O *MongoDB* foi selecionado como banco de dados da aplicação para salvar os dados estatísticos dos jogadores que participam do processo de *Matchmaking*. Foi escolhido um banco de dados não relacional, uma vez que esse permite o armazenamento de dados não estruturados. Trata-se de um requisito, dado que a solução proposta possui critérios variáveis de seleção, e dependentes do jogo em que está sendo aplicado.

3.2.4 Git

O *Git* é uma ferramenta de *software livre* e distribuída para controle de versão, sendo adequada para lidar com projetos de pequena a grande escala (GIT, 2022). O *Git* é utilizado, nesse trabalho, para realizar o versionamento do código fonte do projeto.

3.2.5 GitHub

GitHub é uma plataforma de hospedagem de código-fonte e arquivos, que utiliza o *Git* como controle de versão (GITHUB, 2022). A plataforma é pertinente, nesse trabalho, para hospedar os repositórios remotos que contêm o código-fonte da solução.

3.2.6 Docker

O *Docker* é uma plataforma que oferece a possibilidade de empacotar e executar uma aplicação dentro de um contêiner (DOCKER, 2022). A partir do *Docker*, é possível compartilhar o mesmo contêiner com vários outros desenvolvedores, garantindo com que todos possuam o mesmo ambiente de desenvolvimento.

3.2.7 Node.js

O *Node.js* é um *software* de código aberto que permite a execução de códigos *JavaScript* fora de um navegador da *web* (NODEJS, 2022). Com o *Node.js*, é possível criar códigos para serem executados em linha de comando para auxiliar no desenvolvimento e nos testes da aplicação.

Tabela 1 – Tecnologias Utilizadas para Auxiliar na Pesquisa

Nome	**Descrição**	**Link**	**Versão**
LaTeX	Sistema de preparação de documentos	<https://www.latex-project.org/>	n.a.d
Overleaf	Ferramenta para escrita de documentos científicos	<https://www.overleaf.com/>	n.a.d
Mendeley	Ferramenta de gestão de referências	<https://www.mendeley.com>	n.a.d
Kaggle	Comunidade de Cientistas de Dados	<https://www.kaggle.com/>	n.a.d

Fonte: autores

3.2.8 Jupyter Notebook

O *Jupyter Notebook* é um ambiente de desenvolvimento de software interativo baseado na web. Sua interface permite a configuração e a execução de fluxos de *data science* (JUPYTER, 2022). O *Jupyter Notebook* é usado para criar códigos que auxiliam na análise e na visualização de dados gerados pelo Sistema *Matchmaking*.

3.3 Resumo do Capítulo

Esse Capítulo possui como objetivo descrever as principais tecnologias utilizadas no decorrer do trabalho. Foram apresentadas as Tecnologias de Apoio à Pesquisa, como Latex e Overleaf. Complementarmente, foram abordadas as Tecnologias de Apoio ao Desenvolvimento, como o Jade e sua arquitetura, o RabbitMQ e outros.

A Tabela 1 mostra, brevemente, as tecnologias que auxiliaram no processo de pesquisa do trabalho.

A Tabela 2 mostra, brevemente, as tecnologias que auxiliaram no processo de desenvolvimento do projeto.

Tabela 2 – Tecnologias Utilizadas para Auxiliar no Desenvolvimento

Nome	**Descrição**	**Link**	**Versão**
JADE	*Framework* para Sistemas Multiagente	<https://jade.tilab.com/>	4.5.0
RabbitMQ 3.9.15	Sistema de fila de mensagens	<https://www.rabbitmq.com/>	
MongoDB	Banco de dados baseado em documentos	<https://www.mongodb.com/pt-br>	5.0.0
Git	Ferramenta de controle de versão	<https://git-scm.com/>	2.35.2
GitHub	Plataforma de hospedagem de código-fonte	<https://github.com/>	n.a.d
Docker	Plataforma para empacotamento de aplicações em contêineres	<https://www.docker.com/>	20.10.15
Node.js	Software de execução de linguagem	<https://nodejs.org/en/>	16.14.0
Jupyter Notebook	Ambiente de desenvolvimento	<https://jupyter.org/>	5.0

Fonte: autores

4 Metodologia

Nesse capítulo, tem-se o detalhamento metodológico que orienta a condução do trabalho. Em um primeiro momento, é apresentada a Classificação da Pesquisa, considerando natureza, abordagem, objetivos, e procedimentos. Na sequência, são tratados os Fluxos de Atividades, tanto para TCC_1, quanto para TCC_2. Como algumas dessas atividades são, na verdade, subprocessos, há seções dedicadas às metodologias adotadas em cada subprocesso, sendo: Metodologia de Pesquisa, Metodologia de Desenvolvimento, e Metodologia de Análise de Resultados. Os Cronogramas são cobertos, visando conferir uma noção mais temporal quanto às realizações das atividades. Por fim, há o Resumo do Capítulo.

4.1 Classificação de Pesquisa

Segundo GERHARDT; SILVEIRA (2009), a pesquisa possibilita o entendimento e a aproximação da realidade, sendo um processo que é permanentemente inacabado. As razões para se fazer uma pesquisa científica, podem ser de natureza intelectual, conhecer por querer conhecer, ou de natureza prática, conhecer com o objetivo de aprimorar (GERHARDT; SILVEIRA, 2009).

A pesquisa científica tem por finalidade descobrir respostas para questões utilizando o método científico (FREITAS, 2013). Ela sempre irá partir de um problema, uma questão, um contexto onde o conhecimento disponível não gera a resposta adequada (FREITAS, 2013). Para que seja possível estabelecer metodologias aderentes à presente pesquisa, deve-se classificá-la de acordo com Abordagem, Natureza, Objetivos, e Procedimentos, conforme acordado a seguir.

4.1.1 Abordagem

Quanto à abordagem, pela preocupação que existe em se aprofundar na compreensão sobre uma área do conhecimento dentro do Domínio de Jogos, como Sistemas *Matchmakings*, essa pesquisa pode ser classificada como **híbrida**, tendo particularidades quantitativa e qualitativa, em concordância com GERHARDT; SILVEIRA (2009) e FREITAS (2013). Em termos quantitativos, são analisadas métricas que podem ser mensuradas, tal como tempo levado para uma partida ser formada. Em termos qualitativos, são coletados *feedbacks* orientados a critérios mais subjetivos, tais como a pertinência dos Sistemas Multiagentes para lidar com as necessidades inerentes aos Sistemas *Matchmakings*, ou ainda se a solução proposta é satisfatória e adaptável o suficiente para atender

qualquer jogo.

4.1.2 Natureza

A natureza da pesquisa em questão é **aplicada**, uma vez que a pesquisa visa atingir verdades e interesses locais, dirigindo-se a um problema específico (GERHARDT; SILVEIRA, 2009), no caso, o uso de Sistemas Multiagentes em Sistemas de *Matchmaking* de jogos. Foi desenvolvida uma solução técnica, com foco em um problema específico do domínio dos jogos digitais.

4.1.3 Objetivos

A pesquisa proporciona mais informações sobre o assunto investigado. A baixa compreensão sobre como Sistemas *Matchmakings* podem interagir utilizando-se de Sistemas Multiagentes, faz com que a pesquisa proporcione maior familiaridade com o problema, tornando-o mais explícito e até construindo novas hipóteses. Essa abordagem está de acordo com as pesquisas de viés **exploratório** em confirmação com GIL (2002).

4.1.4 Procedimentos

A primeira parte da pesquisa foi elaborada a partir de materiais já publicados e reconhecidos, portanto uma **pesquisa bibliográfica**. Nesse contexto, foi realizado um levantamento bibliográfico com base em artigos, livros, revistas, e sites. Na pesquisa bibliográfica, é importante os pesquisadores se atentarem à veracidade dos dados obtidos, observando possíveis incoerências (FREITAS, 2013). Outros detalhes sobre a Pesquisa Bibliográfica constam na seção 4.3.

Na segunda parte da pesquisa, são coletados dados do Sistema *Matchmaking* desenvolvido, seguidos de análises, com o intuito dos pesquisadores desempenharem um papel ativo na realidade dos fatos observados (FREITAS, 2013). Dessa forma, é utilizada **pesquisa-ação**. Outras considerações nesse sentido são reportadas na seção 4.5.

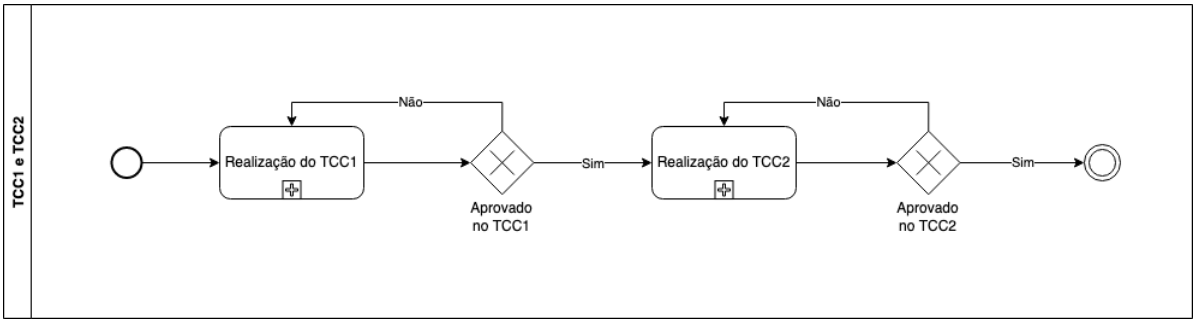
4.2 Fluxos de Atividades

O fluxo geral de atividades para a realização do Trabalho de Conclusão de Curso (TCC) é representado na Figura 3. Nele, podem ser vistos dois subprocessos principais, o TCC1 e o TCC2, que são descritos em detalhes nas Figuras 4 e 5, respectivamente.

4.2.1 Trabalho de Conclusão de Curso

- **Realização do TCC1**: Este subprocesso representa as atividades realizadas durante o Trabalho de Conclusão de Curso 1, considerando o período letivo de 2021.2

Figura 3 – Fluxo Geral de Atividades do TCC



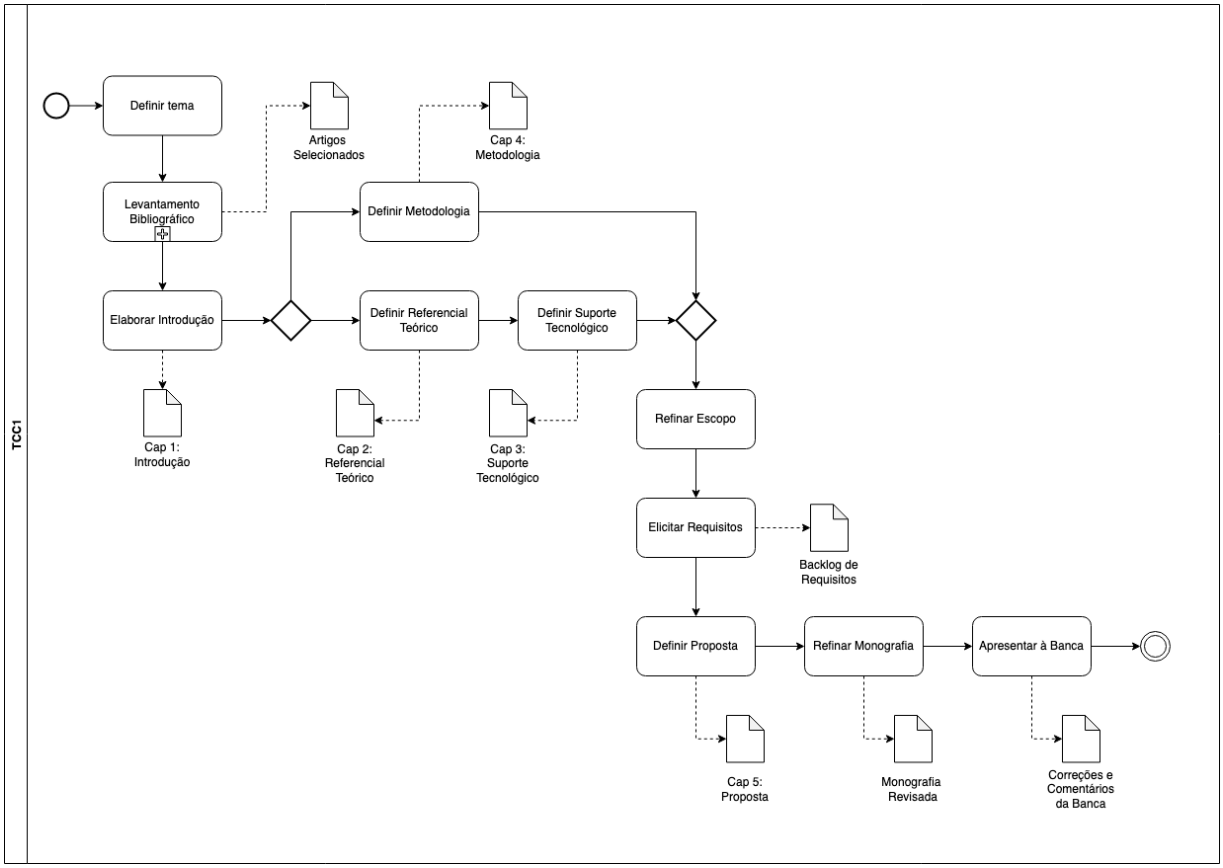
Fonte: autores

O subprocesso pode ser visto em detalhes na Figura 4;

- **Realização do TCC2**: Este subprocesso representa as atividades realizadas durante o Trabalho de Conclusão de Curso 2, considerando o período letivo de 2022.1. O subprocesso pode ser visto em detalhes na Figura 5.

4.2.1.1 Trabalho de Conclusão de Curso 1

Figura 4 – Fluxo de Atividades do TCC1



Fonte: autores

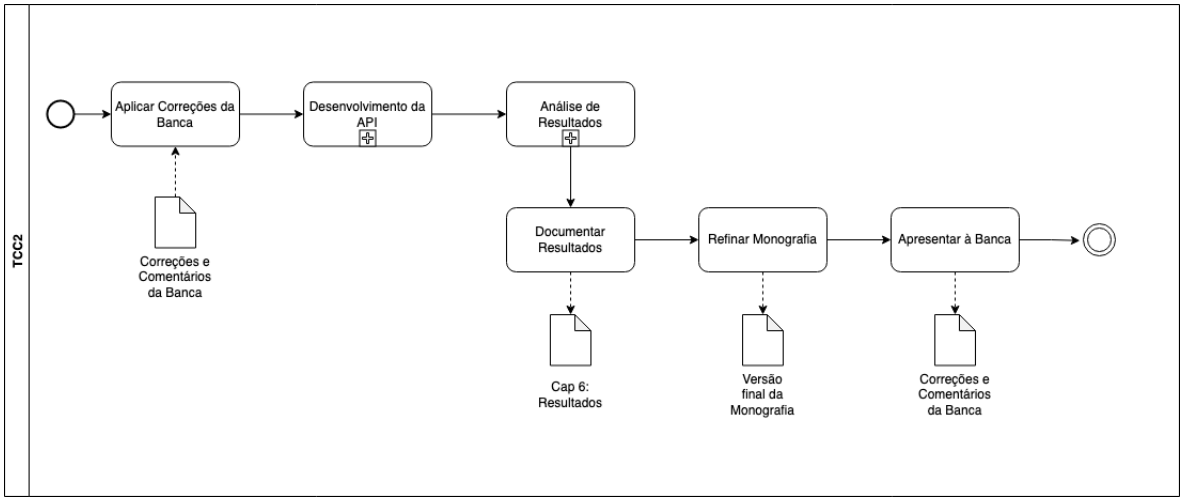
- **Definir Tema**: Esta atividade refere-se à definição da área geral de pesquisa do Trabalho de Conclusão de Curso e outras áreas afins;
- **Levantamento Bibliográfico**: Este subprocesso refere-se à realização do Levantamento Bibliográfico. Teve o objetivo de levantar estudos relacionados à área de pesquisa do trabalho, criando a base de referências que fundamenta o trabalho. Resultado: Capítulo de Referencial Teórico, orientado pela Metodologia descrita na seção 4.3;
- **Elaborar Introdução**: Esta atividade refere-se à escrita do capítulo introdutório do trabalho, com o objetivo de definir e contextualizar a área de pesquisa, além de determinar questão de pesquisa, apoiada em hipóteses; objetivos, e justificativas para o projeto. Resultado: Capítulo de Introdução;
- **Definir Metodologia**: Esta atividade refere-se à definição das metodologias que guiam as etapas de Pesquisa, Desenvolvimento e Análise. Resultados: Metodologias especificadas nas seções 4.3, 4.4 e 4.5, respectivamente;
- **Definir Referencial Teórico**: Esta atividade refere-se à escrita do Capítulo 2, que tem como objetivo sintetizar as fundamentações conceituais, visando o embasamento do trabalho. Resultado: Capítulo de Referencial Teórico;
- **Definir Suporte Tecnológico**: Esta atividade refere-se à definição das tecnologias utilizadas para apoio à pesquisa, além das tecnologias usadas para auxiliar no desenvolvimento do trabalho como um todo. Resultado: Capítulo de Suporte Tecnológico;
- **Refinar Escopo do Trabalho**: Esta atividade refere-se ao refinamento do escopo do trabalho, uma vez que o estudo e o conhecimento dos referenciais, Teórico e Tecnológico, conferiram maior familiaridade com os tópicos de interesse da pesquisa;
- **Elicitar Requisitos**: Esta atividade refere-se ao levantamento de requisitos que a API, orientada a SMA, para mitigar *Matchmaking* no domínio de Jogos Digitais, cumpre. Fez-se uso das técnicas de Elicitação *benchmarking*, observação e anotação. Resultado: seção 5.2;
- **Definir Proposta**: Esta atividade refere-se à escrita da proposta, realizada a partir da compreensão obtida com base nas atividades anteriores. Resultado: Capítulo de Proposta (Monografia de TCC1). É importante notar que o Capítulo de Proposta foi substituído pelo Capítulo de *Multiagents Matchmaking API* ;
- **Refinar Monografia**: Esta atividade refere-se ao refinamento do presente documento, em sua primeira versão (primeira etapa do TCC). O objetivo dessa atividade

é gerar um único artefato, sendo esse responsável por documentar todas as atividades inerentes ao TCC1. Resultado: Própria Monografia (primeira versão), e

- **Apresentar à Banca**: Esta atividade refere-se à última etapa do TCC1. Essa compreendeu a apresentação da proposta à Banca Avaliadora.

4.2.1.2 Trabalho de Conclusão de Curso 2

Figura 5 – Fluxo de Atividades do TCC2



Fonte: autores

- **Aplicar Correções da Banca**: Esta atividade refere-se às correções realizadas na monografia, após as considerações por parte da Banca Avaliadora. Resultado: Monografia Refinada;

- **Desenvolvimento da API**: Este subprocesso refere-se ao desenvolvimento do Sistema *Matchmaking*, guiando-se pela Metodologia estabelecida na seção 4.4. Resultado: API, orientada a SMA, para mitigar *Matchmaking* no domínio dos Jogos *Online*. A API está documentada no Capítulo *Multiagents Matchmaking API*;

- **Análise de Resultados**: Este subprocesso refere-se à análise dos resultados obtidos durante o desenvolvimento do projeto, guiando-se pela Metodologia estabelecida na seção 4.5. Resultado: Insumos coletados e documentados no Capítulo de Análise de Resultados;

- **Documentar Resultados**: Esta atividade refere-se à escrita dos resultados obtidos após a análise de resultados. Resultado: Capítulo Análise de Resultados;

- **Refinar Monografia**: Esta atividade refere-se ao refinamento da monografia. O objetivo dessa atividade é gerar o documento final, com o detalhamento do Trabalho de Conclusão do Curso como um todo. Resultado: Presente Documento, e

- **Apresentar à Banca**: Esta atividade refere-se à última etapa do TCC2. Essa compreende a apresentação do trabalho final à Banca Avaliadora.

4.3 Metodologia de Pesquisa

A pesquisa bibliográfica, como abordada na Seção 4.1.4, foi realizada através do levantamento de artigos, livros, revistas, e sites que pudessem auxiliar no entendimento do domínio abordado no trabalho.

4.3.1 *String* de Busca

A *String* de busca é um conjunto de palavras e operações lógicas que formam uma regra de busca a ser utilizada em bases de dados. Após a definição do tema de pesquisa, parte-se para a elaboração da *string* de busca. Para a obtenção de melhores resultados, no decorrer do processo de pesquisa, há um refinamento da *string*.

A Tabela 3 mostra a evolução da *string* de busca no decorrer da pesquisa, além da quantidade de resultados obtidos em cada base de dados utilizada.

Tabela 3 – Status das Atividades TCC1

String	**Base de Dados**	**Quantidade**
'*matchmaking*' AND '*multiagents*'	IEEE	29
'*game matchmaking*' AND '*multiagents*'	IEEE	1
'*matchmaking*' AND '*multiagents*'	ACM	0
'*game matchmaking*' AND '*multiagents*'	ACM	0
'*game*' AND '*matchmaking*'	IEEE	37
'*matchmaking*' AND '*online game*'	ACM	68

Fonte: autores

É possível notar que houve uma dificuldade de se encontrar conteúdos que ligassem o domínio de *matchmaking* e de multiagentes. Por isso, foi necessário realizar buscas com os domínios separados para se entender melhor a respeito do contexto como um todo.

Além das buscas realizadas nas bases científicas, também foram realizadas pesquisas em sites de notícia e em fóruns *online*, como o REDDIT (2022), com o intuito de entender melhor o contexto de jogos digitais e *matchmaking* em jogos, por meio de comunidades.

4.3.2 Critérios de Seleção

Uma vez que os materiais são coletados, é necessário filtrá-los para que os materiais mais relevantes sejam utilizados no trabalho. Para se realizar essa filtragem, foi utilizada

a leitura exploratória (GIL, 2002). Através disso, considerou-se os seguintes critérios de seleção:

- Tratar algoritmos de *matchmaking*;
- Tratar formas de avaliar as habilidades de um jogador;
- Tratar o impacto do *matchmaking* na experiência do jogo, e/ou
- Tratar o desenvolvimento de sistemas multiagentes.

Com base nisso, foram obtidos alguns artigos, sendo os principais deles:

- *Surrender at 20? matchmaking in league of legends.*
- *Globally optimized matchmaking in online games.*
- *The Rating of Chessplayers, past and present.*
- *TrueskillTM: A bayesian skill rating system, in advances in neural information processing system.*
- *Rule designs for optimal online game matchmaking.*

Além disso, também foram usados, para a pesquisa e aprofundamento no domínio de multiagentes, os seguintes materiais:

- *Developing Multi-Agent Systems with Jade.*
- *Multi-agent systems for power engineering applications - part i: Concepts, approaches, and technical challenges.*

Foi utilizada a ferramenta *Mendeley* (MENDELEY, 2022) para se manter o rastro dos artigos de interesse, além dos insumos contidos em cada um deles. Os embasamentos coletados na etapa de pesquisa foram utilizados no decorrer do projeto, bem como encontram-se documentados nessa monografia.

4.4 Metodologia de Desenvolvimento

Para o desenvolvimento da pesquisa, optou-se por se orientar às práticas ágeis. Por se tratar de um desenvolvimento de um software, mesmo que sendo com viés exploratório, o ágil pode fornecer uma alta capacidade de adaptabilidade ao projeto. Dentro das práticas

ágeis, foi utilizado um sistema híbrido, combinando o **Scrum** (SCHWABER, 2004) e o **Kanban** (AHMAD; MARKKULA; OIVO, 2013).

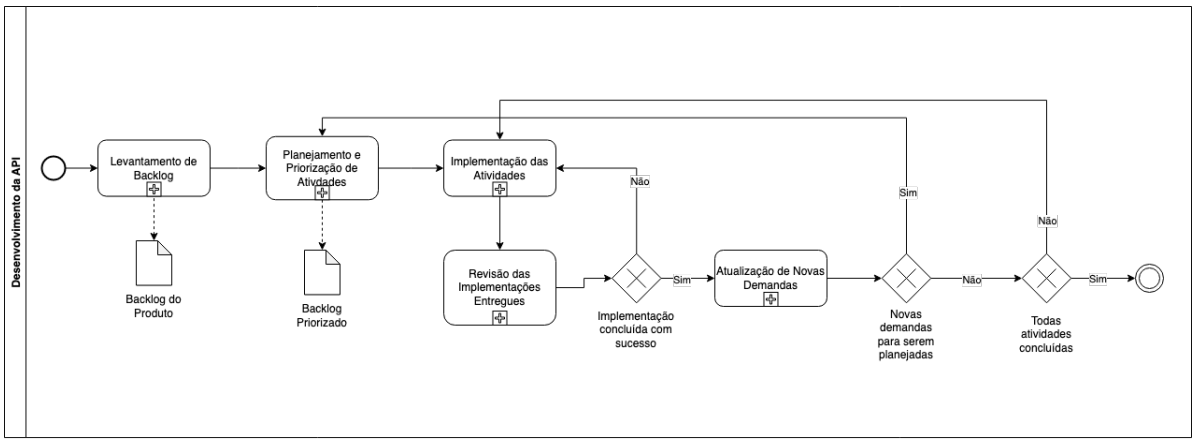
O Scrum coloca todas as suas práticas em um esqueleto incremental e iterativo (SCHWABER, 2004). As atividades são desenvolvidas em iterações e, ao final de cada iteração, tem-se um novo incremento do produto (SCHWABER, 2004). Esse desenvolvimento em iteração permite uma maior flexibilidade sobre possíveis mudanças que o produto pode incorrer, ao longo do seu desenvolvimento. Isso agrega muito ao trabalho, tendo em vista que, por ser uma pesquisa, é necessário entender que os seus resultados podem ser desconhecidos e exigir adequações ao longo do tempo.

Já o Kanban é uma ferramenta de controle de fluxo para produção (AHMAD; MARKKULA; OIVO, 2013), que surgiu nas Fábricas da Toyota. O Kanban, no contexto da Engenharia de Software, surge com a intenção de levar as equipes de desenvolvimento à visualização do fluxo de trabalho; à limitação do Trabalho em Progresso (WIP) em cada estágio do fluxo de trabalho, e ao gerenciamento do tempo do ciclo de trabalho (AHMAD; MARKKULA; OIVO, 2013). O Kanban fornece o poder de observar o todo, e de manter a gestão do conhecimento sobre o andamento do projeto clara para todos os seus participantes.

4.4.1 Fluxo de Desenvolvimento

O Fluxo de Desenvolvimento foi conduzido em etapas, nas quais percebe-se o uso combinado do Scrum com o Kanban, criando um híbrido de ambas as metodologias. O fluxo foi iterativo e incremental, compreendendo as etapas da Figura 6:

Figura 6 – Fluxo de Desenvolvimento da API



Fonte: autores

- **Levantamento de *Backlog***: Nesse subprocesso, foram levantadas todas as tarefas necessárias para poder iniciar o desenvolvimento da pesquisa;

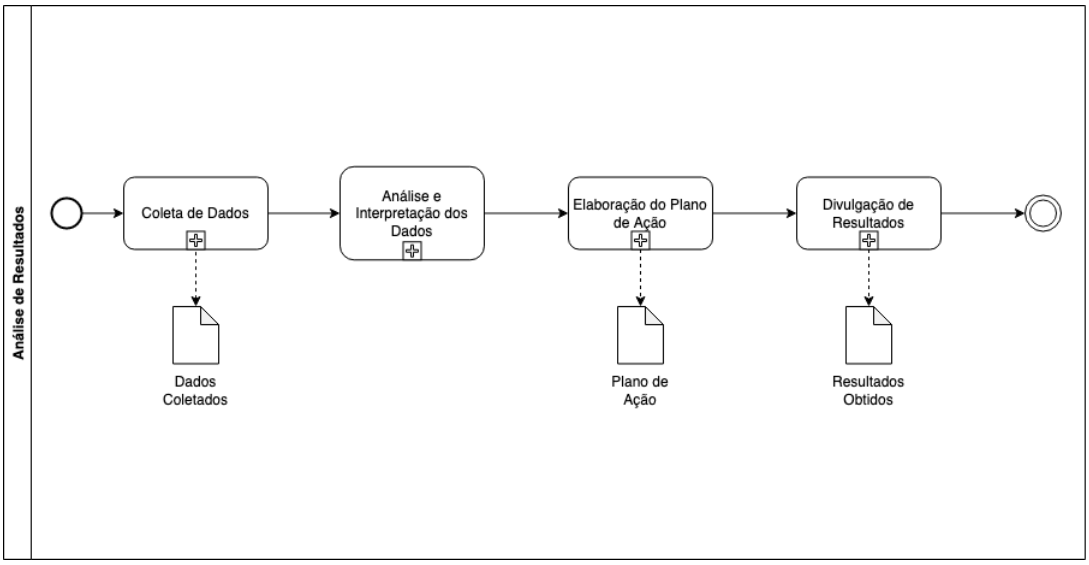
- **Planejamento e Priorização das atividades**: Nesse subprocesso, as tarefas levantadas foram priorizadas de acordo com a sua importância, sendo planejadas para cada nova iteração;
- **Implementação das atividades**: Nesse subprocesso, as tarefas foram realizadas, representando o momento de construção do produto;
- **Revisão das implementações entregues**: Nesse subprocesso, as tarefas que foram finalizadas passam por um processo de revisão e alinhamento para saber se foram concluídas com sucesso, e
- **Atualização de novas demandas**: Quando surgiu a necessidade de modificar ou incrementar novas tarefas, esse subprocesso foi responsável por atualizar uma nova iteração.

4.5 Metodologia de Análise de Resultados

Para a análise de resultados, foi realizada uma **pesquisa-ação**. A Pesquisa-ação envolve a participação ativa dos pesquisadores (GIL, 2002). O planejamento da pesquisa-ação consiste em várias revisitas às atividades anteriores, sendo esses retornos determinados pelos autores da pesquisa (GIL, 2002).

Desse modo, a pesquisa-ação procurou analisar a pertinência do uso de Sistemas Multiagentes em Sistemas de *Matchmaking*. A pesquisa-ação foi orientada pelas etapas da Figura 7:

Figura 7 – Fluxo de Análise dos Resultados



Fonte: autores

- **Coleta de Dados**: Nesse subprocesso, iniciou-se o protocolo de pesquisa-ação. Foram coletados dados de funcionamento do Sistema *Matchmaking* com Multiagentes, com base em métricas quantitativas, bem como *feedbacks* sobre a adaptabilidade da proposta aos diferentes jogos, com base em critérios de qualidade;
- **Análise e Interpretação dos Dados**: Nesse subprocesso, os dados foram analisados e discutidos com a intenção de melhorar a sua interpretação;
- **Elaboração do Plano de Ação**: Nesse subprocesso, foi elaborado um plano responsável por ditar os próximos passos e possíveis melhorias, e
- **Divulgação de Resultados**: Por fim, os resultados foram documentados e divulgados ao final da pesquisa.

4.6 Cronogramas

A Tabela 4 mostra o cronograma de atividades, referente ao TCC1.

Tabela 4 – Cronograma de Atividades TCC1

Atividades	**Jan/2022**	**Fev/2022**	**Mar/2022**	**Abr/2022**	**Mai/2022**
Definir Tema	X				
Levantamento Bibliográfico	X				
Elaborar Introdução	X				
Definir Referencial Teórico		X			
Definir Metodologia		X			
Definir Suporte Tecnológico			X		
Refinar Escopo do Trabalho			X		
Elicitar Requisitos				X	
Definir Proposta				X	
Refinar Monografia				X	
Apresentação à Banca					X

Fonte: autores

A Tabela 5 mostra o cronograma de atividades, referente ao TCC2.

Tabela 5 – Cronograma de Atividades TCC2

Atividades	**Mai/2022**	**Jun/2022**	**Jul/2022**	**Ago/2022**	**Set/2022**
Aplicar Correções da Banca	X				
Desenvolvimento da API	X	X	X		
Análise de Resultados				X	
Documentar Resultados				X	
Refinar Monografia				X	
Apresentação à Banca					X

Fonte: autores

4.7 Resumo do Capítulo

Nesse capítulo, apresentou-se, de maneira detalhada, toda a metodologia da pesquisa.

Foi especificada a classificação da pesquisa, considerando: abordagem, natureza, objetivos e procedimentos. Têm-se: Abordagem Quantitativa e Qualitativa, Natureza Aplicada, Objetivos Exploratórios, e Pesquisa Bibliográfica & Pesquisa-ação, como procedimentos. Depois, foi ilustrado o fluxo de atividades. Esse fluxo foi separado em dois subprocessos de macro nível, responsáveis por representar o TCC 1 e o TCC 2, respectivamente.

Mais adiante, há detalhamentos para cada metodologia específica, sendo: (i) a **Pesquisa Bibliográfica**; (ii) a **combinação de Scrum & Kanban**, com adaptações ao escopo do projeto, como Metodologia de Desenvolvimento, e (iii) a **Pesquisa-ação**, com protocolo estabelecido, como Metodologia de Análise de Resultados. Por fim, os cronogramas são mostrados, tanto para a primeira etapa desse trabalho, quanto para a segunda etapa.

5 *Multiagents Matchmaking API*

Esse capítulo tem o objetivo de esclarecer sobre a aplicação construída durante o Trabalho de Conclusão de Curso, chamada *Multiagents Matchmaking API*. Em um primeiro momento, será apresentada a Contextualização da pesquisa e a aplicação desenvolvida, com as suas motivações e os seus detalhamentos. Na sequência, constam os Requisitos da Aplicação, com o *backlog* priorizado das tarefas realizadas durante o desenvolvimento. Depois, descreve-se a Arquitetura do sistema desenvolvido, apresentando a Arquitetura Geral, os Agentes inteligentes, o seu Sistema de Comunicação, as Classes e o Sistema de Recuperação de Falhas desenvolvidos durante o trabalho. Mais adiante, há uma breve apresentação de como ocorreu o desenvolvimento do trabalho, orientando-se pela Metodologia de Desenvolvimento utilizada. Por fim, tem-se o Resumo do Capítulo.

5.1 Contextualização

Com o surgimento da pandemia da Covid-19, governos no mundo inteiro foram forçados a estabelecer medidas de distanciamento social, o que impossibilitou as pessoas levarem suas vidas da forma como levavam antes. Permanecendo mais tempo em casa, bem como sem poder interagir diretamente com seus amigos, a popularidade de plataformas de mídias digitais aumentou entre as pessoas. Dentre essas plataformas, pode-se mencionar: redes sociais, redes de video-chamadas, e jogos *online*.

Os jogos *online* cresceram muito nos últimos anos. Relatórios da Verizon (VERIZON, 2022) mostram que o uso de tráfico de rede para jogos *online* cresceu em até 82%, em comparação com os níveis pré-pandemia nos Estados Unidos (SCHULZ, 2020). Os jogos como serviço (GaaS), ou jogos que são gratuitos para jogar (*free-to-play*), foram alguns dos maiores responsáveis por popularizar ainda mais os jogos multijogadores, aumentando a sua base de jogadores e, consequentemente, também aumentando a quantidade de jogos desse gênero sendo desenvolvidos.

Entretanto, um aspecto desses jogos *online* é motivo de discussão em diversos fóruns pela internet, o *Matchmaking* (RIOTGAMES, 2021). As várias reclamações compreendem vários jogos, nos quais os jogadores reclamam desde partidas desbalanceadas (HALO, 2021), até alta demora para formação das partidas (LEGENDS, 2020). Além disso, Sistemas *Matchmakings* não são tão fáceis de serem desenvolvidos, despendendo grande tempo de seus desenvolvedores. Esse tempo poderia ser utilizado, por exemplo, para refinar o jogo.

Com base nos pontos acordados anteriormente, foi levantado um primeiro questio-

namento: "*Como criar um Sistema de Matchmaking genérico o suficiente para ser utilizado por qualquer gênero de jogo?*". Depois, surgiu a questão de pesquisa, que de fato motivou esse trabalho, sendo: Sistemas Multiagentes são aderentes para o desenvolvimento de Sistemas *Matchmaking* de Jogos *Online*? Com essas dúvidas em mente, foi observado que o uso de cognição poderia contribuir para esse sistema lidar com diferentes tipos de gênero de jogos, especializando o *Matchmaking*.

Pensando no uso de entidades inteligentes, cognitivas, considerou-se o uso de Sistemas Multiagentes. Esse paradigma orienta-se por um modelo conceitual apoiado em autonomia, colaboração/competição, adaptabilidade, dentre outros princípios. Os Agentes autônomos podem ou não possuir os mesmos objetivos, individualmente, falando. Entretanto, em um SMA, trabalham de forma colaborativa, interativa, em prol de um objetivo. A ideia é usar essa cooperação, dividindo o problema de *Matchmaking* em tarefas menores e complementares, e permitindo a implementação de Agentes cognitivos bem focados em cada tarefa. Ao concluírem essas tarefas, tem-se a solução do problema original.

Visando responder a pergunta levantada, e percebendo a aparente pertinência dos Sistemas Multiagentes no contexto, optou-se pela construção de algo concreto, associando ambos os domínios, Sistemas *Matchmaking* e Sistemas Multiagentes, resultando, assim, na construção de uma API de Sistema de *Matchmaking* genérica, utilizando Multiagentes para a classificação dos jogadores. A API é tratada nesse trabalho como: *Multiagents Matchmaking API*.

5.2 Requisitos da *Multiagents Matchmaking API*

Para a elicitação dos requisitos da API desenvolvida, foram utilizadas as técnicas de *benchmarking*, observação e anotação. Com as três técnicas, foram analisados produtos similares, e o funcionamento de Sistemas *Matchmaking* em jogos que já estão no mercado.

A partir da elicitação dos requisitos, foi especificado um *backlog* de produto. O *backlog* é uma lista com breves descrições das funcionalidades desejadas do produto (SCHWABER, 2004). Após a escrita do *backlog*, as funcionalidades foram priorizadas, utilizando-se uma técnica chamada MoSCoW. Trata-se de uma técnica de priorização, a qual auxilia os *stakeholders* obterem um entendimento comum em relação às funcionalidades de um produto.

A Figura 8 representa o *backlog* priorizado da API desenvolvida.

Figura 8 – *Backlog* Priorizado da API

Épico		Feature		Prioridade
EP01	Gerência de Jogadores	FE01	O Jogador poderá entrar na fila a qualquer momento	Must
		FE02	O Jogador poderá sair na fila a qualquer momento	Should
		FE03	Os Jogadores que estão a mais tempo na fila devem ter prioridade para entrar nos lobbies	Could
		FE04	Os Jogadores devem ser removidos da fila caso sejam adicionados a um lobby	Must
		FE05	O Jogador não pode entrar na fila caso esteja vinculado a um lobby	Should
EP02	Matching de Jogadores	FE06	Um Critério deve ter nome, valor mínimo, valor máximo e valor de prioridade	Must
		FE07	Os Desenvolvedores devem poder adicionar N Critérios para o *Matchmaking*	Should
		FE08	Os Desenvolvedores devem adicionar no mínimo um Critério	Must
		FE09	O Sistema deve salvar os dados estatísticos dos jogadores	Would
		FE10	O Sistema deve atualizar os dados estatísticos dos jogadores quando o jogador entrar na fila	Would
		FE11	Os Desenvolvedores devem ser capazes de informar a quantidade de times e a quantidade de jogadores por time em uma partida	Must
		FE12	O Sistema deve selecionar Jogadores com base na quantidade de times, quantidade de jogadores por time e Critérios adicionados pelos Desenvolvedores	Must
		FE13	O Sistema deve avaliar se os times são considerados balanceados utilizando os Critérios	Must
EP03	Gerencia de Lobbies	FE14	O Sistema deve ser configurado com a quantidade de Lobbies que devem ser organizados por vez	Should
		FE15	O Sistema deve retornar uma resposta informando qual lobby foi formado	Would
		FE16	O Sistema deve salvar quais jogadores estavam vinculados a qual lobby no banco de dados	Would
		FE17	Lobbies com times não balanceados não devem ser vinculados a um servidor	Should

Fonte: autores

No *backlog*, são abordados alguns termos referentes ao domínio, sendo eles:

- **Jogador**: diz respeito à pessoa que está jogando o jogo que utiliza um sistema de *matchmaking* baseado na API;
- **Fila**: diz respeito ao conjunto de jogadores que pretendem jogar o jogo, levando em consideração a ordem em que pediram para jogar;
- **Lobby**: diz respeito ao conjunto de jogadores que formam um grupo que satisfaz os requisitos de uma partida;
- **Critério**: diz respeito a um valor, que pode representar uma habilidade, nível ou uma métrica dentro do jogo;
- **Desenvolvedores**: diz respeito ao grupo de pessoas que utiliza a API no desenvolvimento do seu sistema de *matchmaking*, e
- **Sistema**: diz respeito ao sistema de *matchmaking* que foi implementado utilizando a API.

5.3 Arquitetura

A Arquitetura de um software é fundamental para esclarecer os requisitos de um sistema, sejam eles funcionais ou não funcionais. Uma boa Arquitetura é importante. Caso contrário, torna-se mais complicado adicionar novos recursos, futuramente, no software (FOWLER, 2019).

A definição de Arquitetura não é simples, e diversos autores propõem diversas definições. Ralph Johnson (GAMMA et al., 1994), em uma troca de emails com Martin Fowler (FOWLER, 2003), definiu Arquitetura como sendo: *"Arquitetura é sobre as coisas importantes. Seja o que for"*. Trata-se de uma definição um tanto quanto vaga, e que confere um cenário meio caótico, impreciso. Entretanto, acorda sobre a relevância de uma boa Arquitetura.

No contexto do presente trabalho, a Arquitetura de Software foi desenvolvida para estudar e compreender melhor sobre Sistemas *Matchmaking*. Foi adicionada uma complexidade a mais, a qual demanda compreender como Sistemas Multiagentes podem atuar nesse contexto.

Após o levantamento do *backlog* do produto, foi feita uma análise, tendo esse artefato como insumo. Como resultado, foram elencadas as principais funcionalidades, bem como especificada a Arquitetura de Software.

5.3.1 Arquitetura Geral

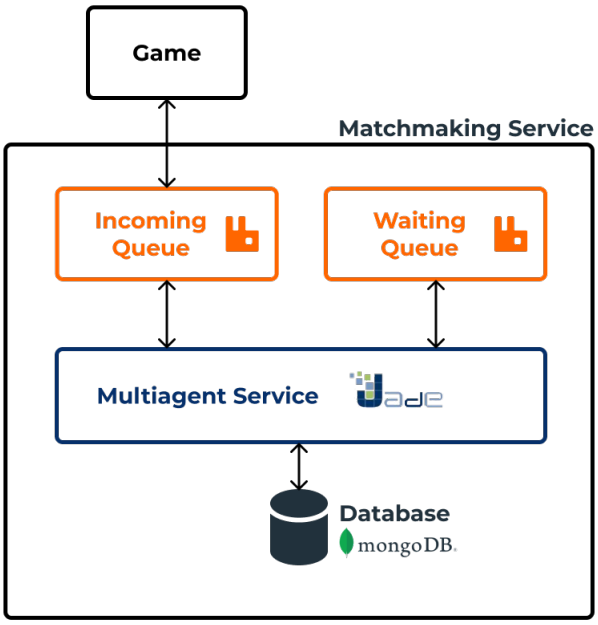
Na Figura 9, é possível ver a Arquitetura da *Multiagents Matchmaking API*.

Dentre os componentes arquiteturais, há um serviço de mensageria. O serviço de mensageria tem o intuito de enfilerar os jogadores que desejam entrar em uma partida, e tornar toda a comunicação para a construção do *Matchmaking* assíncrona (TULIO, 2020). A necessidade de uma comunicação assíncrona ocorre, devido ao fato do jogo não poder aguardar a resposta de que uma partida foi montada. Além disso, caso o *Multiagent Service* fique indisponível, o serviço de mensageria pode manter as mensagens salvas, até o *Multiagent Service* voltar a operar (TULIO, 2020).

O jogo comunica-se com o serviço de *Matchmaking* através desse serviço de mensageria, representado pela *Incoming Queue*. A *Incoming Queue* é responsável por centralizar a informação dos jogadores que desejam participar de um jogo. Além da *Incoming Queue*, também há outra fila de mensagens, a *Waiting Queue*, responsável por armazenar os jogadores que já foram processados, mas não foram escolhidos por nenhum *lobby*, sendo esse um agente de *software* que será apresentado mais adiante.

Outro componente arquitetural é o banco de dados. O intuito do uso de um banco de dados não relacional, orientado a documentos, é justamente para permitir que um

Figura 9 – Arquitetura Geral da Aplicação



Fonte: autores

Matchmaking possa escalar para diferentes gêneros de jogos. Como os jogos podem ser diferentes, é difícil definir uma *Schema* padrão para cada jogo, tornando-se necessário salvar os dados de maneira genérica, tratando-os à medida que forem necessários (SADALAGE; FOWLER, 2013).

Por fim, o *Multiagent Service* é responsável por montar a lógica de pareamento dos jogadores. O serviço foi construído com base no Jade (JADE, 2022). O Jade confere apoio à construção de um Sistema Multiagente. Esse serviço é responsável por receber as mensagens dos serviços de mensageria, e fazer as negociações para o pareamento dos jogadores. Com uma negociação encerrada, uma sala de jogadores é gerada.

5.3.2 Arquitetura Orientada ao Sistema Multiagentes

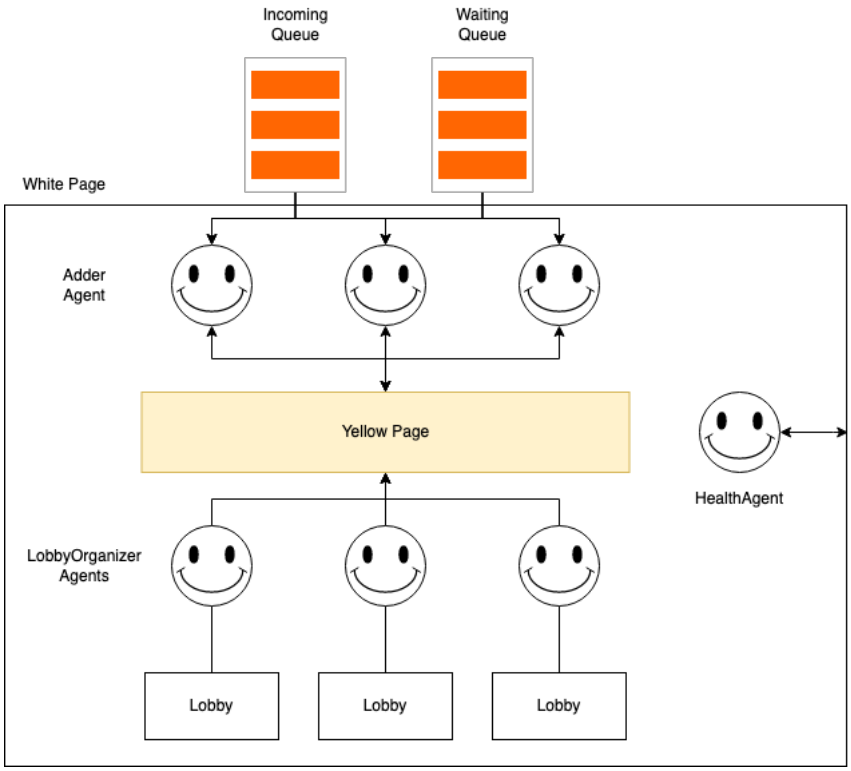
A camada da arquitetura dedicada ao *"Multiagent Service"* representa o Sistema Multiagentes que possui toda a lógica e a cognição do *Matchmaking*. Essa camada possui uma arquitetura de comunicação própria.

A Figura 10 é uma representação da arquitetura de comunicação do Serviço de Multiagentes.

O sistema é composto por três tipos diferentes de agentes, sendo eles:

- ***AdderAgent***: esse agente pode ter de 1 a n instâncias, sendo cada uma delas responsável por fazer a leitura da *Incoming Queue* e *Waiting Queue* e oferecer os

Figura 10 – Arquitetura de Comunicação dos Agentes



Fonte: autores

jogadores presentes nas filas aos *LobbyOrganizerAgents*;

- ***LobbyOrganizerAgent***: esse agente pode ter de 1 a n instâncias, sendo cada uma delas responsável por gerenciar os *lobbies* de jogadores. Ao receber uma oferta de jogador de um *AdderAgent*, o *LobbyOrganizerAgent* avalia se deseja ficar com ele e retorna uma oferta ao *AdderAgent*, e
- ***HealthAgent***: esse agente só apresenta 1 instância, sendo responsável por verificar o estado dos demais agentes. Caso julgue necessário, pode instanciar novos agentes.

Além dos agentes, a arquitetura apresenta duas novas entidades, sendo elas:

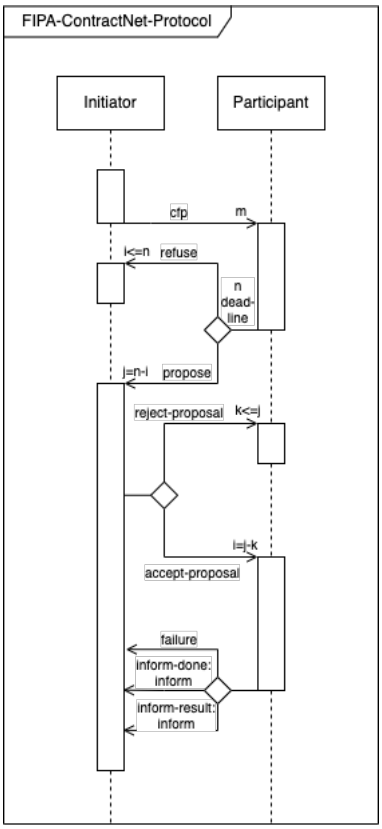
- ***Yellow Page***: entidade onde os agentes registram que estão disponíveis para realizar ações. Cada agente registra-se na *Yellow Page* ao ser instanciado, e
- ***White Page***: essa entidade é capaz de ver quais agentes estão funcionando e qual o estado de cada um deles.

5.3.2.1 Comunicação entre Agentes

Para abordar o sistema de comunicação dos Agentes, precisa-se, primeiramente, abordar o Protocolo *Contract Net* da FIPA. O *Contract Net* é um protocolo de compartilhamento de tarefas em Sistemas Multiagentes. Ele é utilizado para alocar tarefas entre

agentes autônomos (FIPA, 2002b). Na Figura 11, é possível ver um fluxo seguindo esse protocolo.

Figura 11 – Protocolo FIPA Contract Net



Fonte: FIPA (2002b)

O *Contract Net* segue as seguintes etapas:

1. O protocolo é iniciado por um iniciador, que envia uma mensagem do tipo CFP (*Call for Proposal*) para os participantes;
2. Os participantes podem responder com uma mensagem ou do tipo *Refuse*, recusando a proposta, ou do tipo *Propose*, aceitando a proposta;
3. O iniciador escolhe entre os participantes o que apresentou a melhor resposta e envia uma mensagem do tipo *Accept Proposal*, para o participante que apresentou a melhor resposta, e do tipo *Reject Proposal*, para os demais participantes, e
4. Quando o contrato foi estabelecido, o participante escolhido envia uma mensagem do tipo *Inform*. Caso necessário, um resultado pode ser enviado pela mensagem. Caso o participante não consiga cumprir com o acordo, ele envia uma mensagem do tipo *Cancel*.

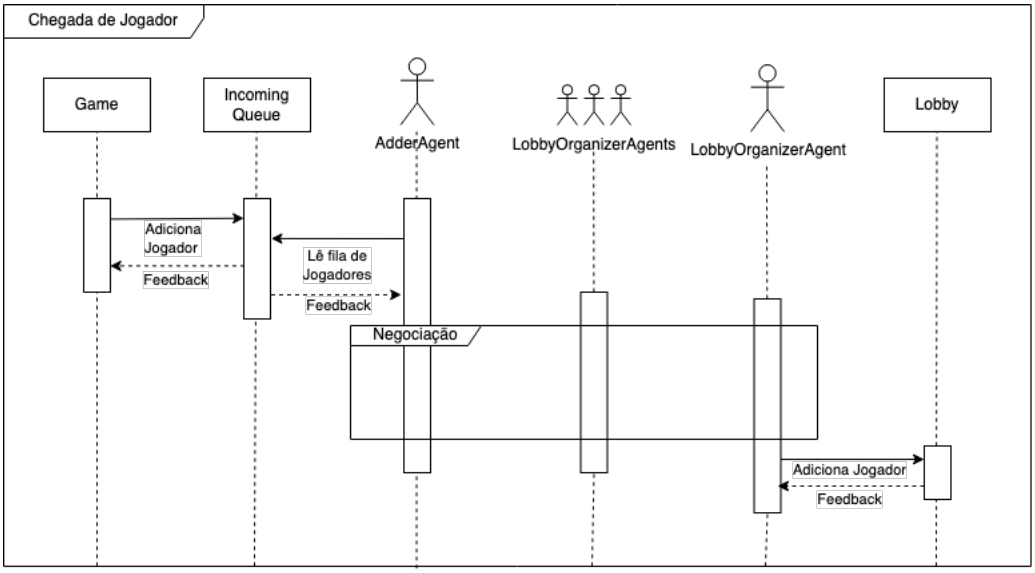
A comunicação entre os Agentes e a forma com que eles interagem para cumprir ações necessárias no sistema são aspectos muito importantes, os quais são abordados

nessa seção. Existem três ações principais, que utilizam o Protocolo *Contract Net*, que devem ser levadas em consideração durante o processo de *matchmaking* da *Multiagents Matchmaking API*, sendo elas: **a entrada de um Jogador na fila; a negociação de um Jogador, e o fechamento de um *Lobby***.

Entrada de Jogador na Fila:

A entrada dos Jogadores na fila acontece quando o jogo envia uma mensagem para a *Incoming Queue*. Essa mensagem contém os dados de um jogador que gostaria de participar de uma partida. Enquanto isso, um *AdderAgent* estará observando a fila para ver se há um novo jogador. Caso tenha algum jogador na fila, o *AdderAgent* inicia um processo de Negociação de Jogador. No final da negociação, um *LobbyOrganizerAgent* adiciona o jogador negociado ao seu *lobby*. Esse comportamento é descrito através de um Diagrama de Sequência na Figura 12.

Figura 12 – Fluxo de Entrada do Jogador na Fila



Fonte: autores

O trecho de código, referente à entrada do jogador na fila, faz parte da lógica do *AdderAgent*, e pode ser visto na Figura 13, que representa o código de adição do comportamento ao agente, e na Figura 14, que descreve o método utilizado.

Figura 13 – Código de Adição do Comportamento de Entrada de Jogadores na Fila

```
36      addBehaviour(new TickerBehaviour( a: this,  period: 2000) {
37          @Override
38          protected void onTick() {
39              getPlayerFromQueueAndOffer(incomingQueueListener);
40          }
41      });
```

Fonte: autores

Figura 14 – Código do Método de Entrada de Jogadores na Fila

```
58 @    private void getPlayerFromQueueAndOffer(AmqpListener queueListener) {
59          QueueMessage queueResponse = queueListener.getMessage();
60          if (queueResponse != null) {
61              String message = queueResponse.getMessage();
62              if (message != null) {
63                  Player playerToAdd = JsonParser.entity(message, Player.class);
64                  if (playerToAdd != null) {
65                      System.out.println(getLocalName() + " offer player!");
66                      addBehaviour(new OfferPlayerBehaviour(playerToAdd, queueListener, queueResponse.getDeliveryTag()));
67                  }
68              }
69          }
70      }
```

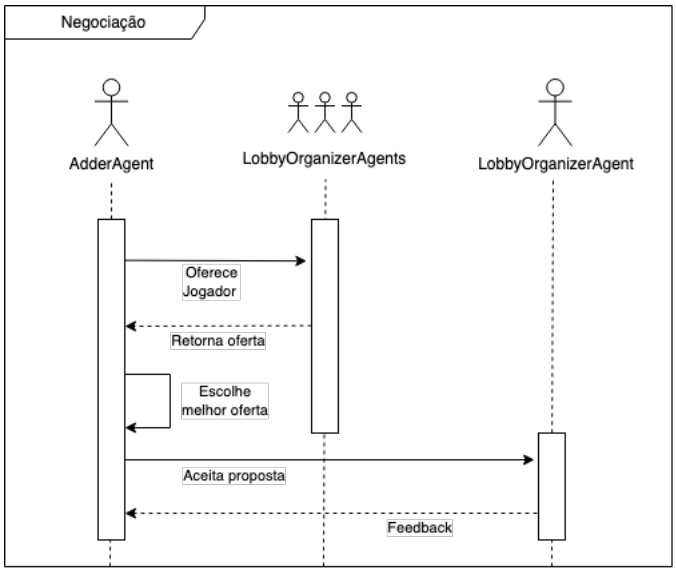
Fonte: autores

Negociação de Jogador:

O processo de negociação faz parte do processo de Entrada de Jogador na Fila.A negociação é o principal ponto de contato entre os Agentes, além de ser o ponto que contém as principais lógicas do *Matchmaking*.

A negociação inicia-se quando um *AdderAgent* oferece um jogador. Cada um dos *LobbyOrganizerAgents* é responsável por responder com uma oferta. Então, o *AdderAgent* irá decidir qual das ofertas é a mais adequada, aceitando a proposta do *LobbyOrganizerAgent* que enviou a melhor oferta. A sequência de ações do processo de negociação é descrita na Figura 15.

Figura 15 – Fluxo de Negociação de Jogador

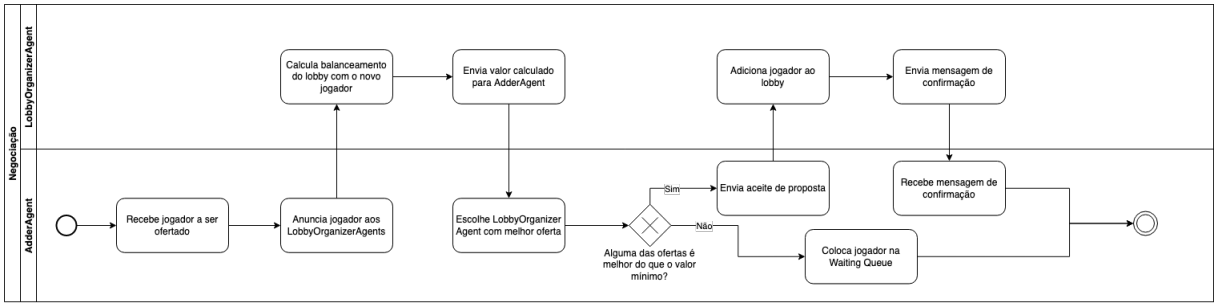


Fonte: autores

Na Figura 16, tem-se o fluxo de negociação do jogador. Esse fluxo possui uma etapa de decisão que determina se o jogador será enviado a um lobby ou a *Waiting Queue*.

O cálculo do balanceamento do *lobby* segue o seguinte processo:

Figura 16 – BPMN da Negociação de Jogador



Fonte: autores

- Primeiro, calcula-se a média simples (m_x) de cada critério do jogo, seguindo a Equação 5.1. Sendo n a quantidade de pessoas no *lobby* e x_i o valor do critério de um jogador i;
- Depois, calcula-se o desvio padrão (s_x) de cada um dos critérios do jogo, seguindo a Equação 5.2;
- O mesmo processo descrito nos itens anteriores é realizado novamente, porém agora com o jogador em negociação adicionado ao *lobby*. O desvio padrão dessa segunda versão do *lobby* é representado como s_y;
- Então, calcula-se a diferença de s_x e s_y, resultando em Δ_j, que representa a diferença do desvio padrão de um critério j. O calculo realizado pode ser visto na Equação 5.3, e
- Por último, calcula-se a média simples de Δ_j, representado por m_Δ como mostra a Equação 5.4. Sendo o m_Δ resultante o *score* de balanceamento do *lobby*.

$$m_x = \frac{x_1 + x_2 + x_3 + ... + x_n}{n} \tag{5.1}$$

$$s_x = \sqrt{\sum_{i=1}^{n} \frac{(x_i - m_x)^2}{n-1}} \tag{5.2}$$

$$\Delta_j = s_x - s_y \tag{5.3}$$

$$m_\Delta = \frac{\Delta_1 + \Delta_2 + \Delta_3 + ... + \Delta_n}{n} \tag{5.4}$$

O trecho de código referente ao processo de negociação pode ser visto na Figura 17.

Figura 17 – Código do Processo de Negociação

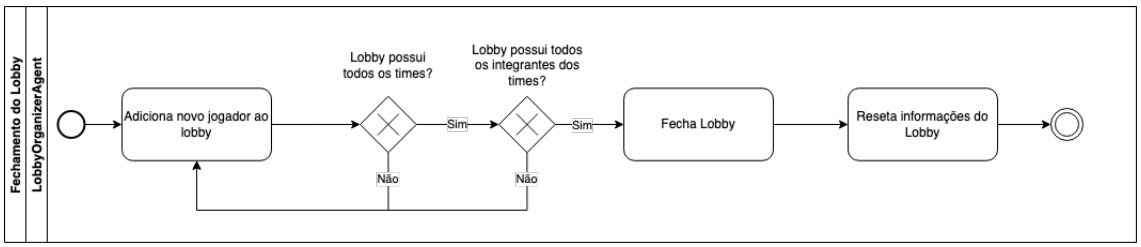
```
101     @Override
102     public void action() {
103         switch (actionStep) {
104             case 0:
105                 _offerPlayer();
106                 break;
107             case 1:
108                 _waitForProposals();
109                 break;
110             case 2:
111                 _acceptProposal();
112             case 3:
113                 _waitForConfirmation();
114         }
115     }
```

Fonte: autores

Fechamento de *Lobby*

O fechamento de um *lobby* ocorre quando o mesmo cumpre todos os critérios de quantidade de times e quantidade de jogadores estabelecidos pelos desenvolvedores do jogo. Essa verificação ocorre pelo *LobbyOrganizerAgent* sempre que um novo jogador é adicionado ao *lobby*. No caso de estarem completos, o *LobbyOrganizerAgent* fecha o *lobby* e reinicia o *lobby* sem informação alguma, para que possa preenchê-lo novamente. A tomada de decisão para verificar se o *lobby* está fechado é descrita na Figura 18.

Figura 18 – BPMN de Fechamento de *Lobby*



Fonte: autores

O código referente ao fechamento do *lobby* pode ser visto na Figura 19.

5.3.3 Classes

Além dos Agentes descritos na Seção 5.3.2, a *Multiagents Matchmaking API* também é constituída por um conjunto de classes auxiliares, como pode ser visto na Figura 20, que mostra a árvore de arquivos da aplicação.

Figura 19 – Código de Fechamento de *Lobby*

```
173             if (completedTeams >= queueConfig.teamAmount) {
174                 for (List<Player> team : lobby.getLobby()) {
175                     System.out.println(myAgent.getName() + " completed Team in Lobby: " + team);
176                 }
177                 try {
178                     _setLobbyFinalTime();   // Set final time on Lobby and save history
179                 } catch (Exception e) {
180                     System.out.println(e.getCause());
181                     System.out.println(e.getLocalizedMessage());
182                     for (StackTraceElement element : e.getStackTrace()) {
183                         System.out.println(element);
184                     }
185                 }
186                 resetLobby();
187             }
```

Fonte: autores

Os módulos apresentados na pasta *config* possuem as respectivas responsabilidades:

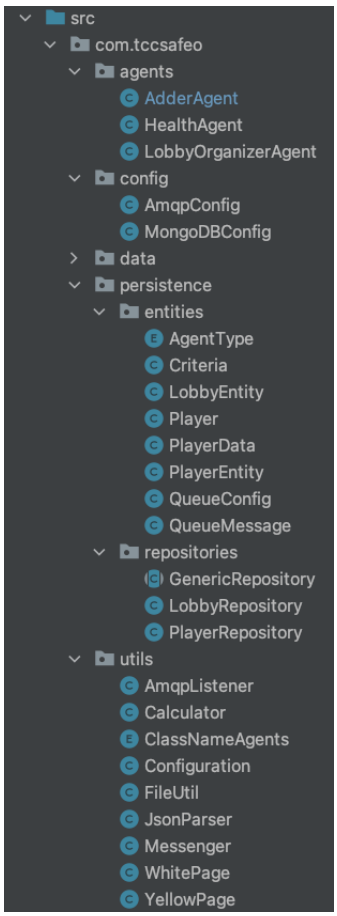
- ***AmqpConfig***: configurar a aplicação para utilizar as filas de mensagem, feitas utilizando RabbitMQ, e
- ***MongoDBConfig***: configurar a aplicação para utilizar o banco de dados, feito utilizando MongoDB.

Os módulos apresentados na pasta *persistence/entities* possuem como responsabilidade definir as entidades que são utilizadas na aplicação. Já os módulos apresentados na pasta *persistence/repositories* possuem como responsabilidade estabelecer ações entre a aplicação e o banco de dados.

Há também uma pasta de classes utilitárias, sendo que elas possuem as seguintes responsabilidades:

- ***AmqpListener***: estabelecer métodos para auxiliar na interação com as filas do RabbitMQ;
- ***Calculator***: estabelecer métodos para auxiliar em cálculos utilizados pela aplicação;
- ***ClassNameAgents***: facilitar o processo de nomear agentes;
- ***Configuration***: salvar as configurações da aplicação globalmente;
- ***FileUtil***: facilitar o manuseio de arquivos;

Figura 20 – Árvore de Arquivos



Fonte: autores

- ***JsonParser***: facilitar questões de *parser* de Json;
- ***Messenger***: estabelecer métodos para envios de mensagens entre agentes;
- ***WhitePage***: estabelecer métodos para facilitar a utilização de *White Pages* pelos agentes, e
- ***YellowPage***: estabelecer métodos para facilitar a utilização de *Yellow Pages* pelos agentes.

5.3.4 Tolerância e Recuperação de Falhas

Um dos principais benefícios de se utilizar Sistemas Multiagentes é a capacidade de recuperação de falhas, ou tolerância a falhas. Isso ocorre, pois dentro da API existem vários agentes capazes de realizar a mesma ação, por várias instâncias funcionarem simultaneamente e com as mesmas responsabilidades, o sistema se mostra tolerável a falhas, dado que o sistema continuará funcionando, mesmo que em uma velocidade menor, caso um agente pare de funcionar adequadamente.

Na *Multiagents Matchmaking API*, existe também uma forma de se evitar falhas. Sendo ela na interação entre o sistema *Matchmaking* e o jogo. Para evitar que ocorra uma sobrecarga de processamento proveniente do jogo, é utilizada uma fila de mensageria para gerir as mensagens enviadas pelo jogo. Dessa forma, o sistema *Matchmaking* processa apenas a quantidade de mensagens que é capaz de lidar. Enquanto isso, as demais mensagens ficam esperando na fila.

Já relacionado à recuperação de falhas, a aplicação é capaz de instanciar vários *AdderAgents* e *LobbyOrganizerAgents*. Dessa forma a aplicação continuará funcionando, caso um deles pare de funcionar. Para lidar com a possível queda de agentes, a comunicação entre vários agentes utiliza um sistema de *timeout*. Assim, o agente irá prosseguir com suas operações, caso receba todas as respostas, ou caso o tempo limite de resposta seja ultrapassado. É possível ver, na Figura 21, um trecho de código que utiliza esse sistema de *timeout*.

Figura 21 – Código com Sistema de *Timeout*

```
140     private void _waitForProposals() {
141         System.out.println("Time: " + (System.currentTimeMillis() - executionStart));
142         if (System.currentTimeMillis() - executionStart > 30000) {
143             actionStep++;
144         } else {
145             reply = myAgent.receive(mt);
146             if (reply != null) {...} else {
159                 block();
160             }
161         }
162     }
```

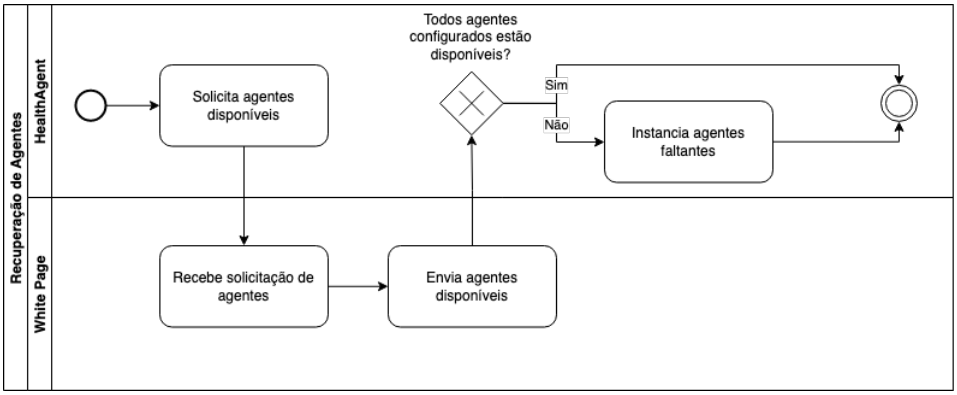
Fonte: autores

Há também um agente responsável apenas por recuperar agentes que não estão funcionamento corretamente, o *HealthAgent*. O *HealthAgent* verifica, na *White Page*, quais agentes estão disponíveis, e compara com os agentes que foram inicialmente instanciados na API. Caso o número de agentes disponíveis seja diferente do número de agentes inicialmente instanciados, o *HealthAgent* cria uma nova instância do agente que está faltando. Dessa forma, a aplicação consegue manter a quantidade desejada de agentes mesmo que ocorra uma falha. Esse processo de recuperação pelo *HealthAgent* está descrito na Figura 22.

5.3.5 Adaptabilidade do Sistema

Um dos critérios avaliados durante a realização desse trabalho é a adaptabilidade desse sistema para que possa ser usado em qualquer jogo. Para alcançar esse objetivo, foram determinados três critérios que poderiam ser usados para se configurar o sistema

Figura 22 – BPMN de Recuperação de Agentes



Fonte: autores

para a maioria dos jogos, sendo eles: Quantidade de times; Quantidade de jogadores por time e Critérios dos jogadores que devem ser avaliados.

Na Figura 23, pode-se ver um exemplo do arquivo de configuração que deve ser usado na *Multiagent Matchmaking API*.

Figura 23 – Arquivo de Configuração

```
{
  "criteria": [
    { "name": "Squads score", "min": 1394, "max": 5275367 },
    { "name": "Squads top1", "min": 199, "max": 5116 },
    { "name": "Squads kd", "min": 0.94, "max": 10.24 },
    { "name": "Squads winRatio", "min": 3.2, "max": 39.3 },
    { "name": "Squads matches", "min": 4, "max": 28805 },
    { "name": "Squads kills", "min": 4588, "max": 117327 },
    { "name": "Squads minutesPlayed", "min": 74, "max": 150439 }
  ],
  "teamSize": 4,
  "teamAmount": 25
}
```

Fonte: autores

5.4 *Scripts* Auxiliares

Além da API, também foram providos códigos de *scripts* para auxiliar nos testes e na análise dos resultados obtidos.

5.4.1 Formatador de Dataset

Foi feito um *script* utilizando *Node.js* para formatar bases de dados em formato .csv para o formato utilizado pela API desenvolvida. Assim, facilitando a utilização de

diferentes bases de dados para testes na API desenvolvida.

Na Figura 24, é possível ver um trecho do código que formata o csv de jogadores para o formato correto.

Figura 24 – Código Formatação de *Dataset*

```
22  function formatData(row) {
23    const playerData = [];
24
25    for (let i = 1; i < columnNames.length; i++) {
26      playerData.push({
27        key: columnNames[i],
28        value: parseFloat(row[i]),
29      });
30    }
31
32    data.push({
33      playerId: row[0],
34      playerData,
35    });
36  }
```

Fonte: autores

5.4.2 Gerador de Configuração

Adicionalmente, foi feito um *script* utilizando *Node.js* para gerar um arquivo de configuração que pode ser utilizado pela *Multiagents Matchmaking API*, facilitando o processo de configuração do sistema *Matchmaking*.

Na Figura 25, é possível ver um trecho do código que cria um arquivo de configuração para ser utilizado pela API a partir do .csv lido.

5.4.3 Envio de Jogadores

Consta ainda um *script* utilizando *Node.js* para simular o envio de mensagens para a *Incoming Queue*. Como o desenvolvimento de um jogo não faz parte do escopo desse Trabalho de Conclusão de Curso, foi desenvolvido esse *script* que envia informações de jogadores para a fila de mensagens.

Na Figura 26, é possível ver um trecho do código que realiza esse envio de jogadores.

5.4.4 Código de Análise de Dados

Complementarmente, há também um código utilizando um Jupyter Notebook que se conecta ao banco de dados da *Multiagent Matchmaking API*, organiza os dados cole-

Figura 25 – Código Gerador de Configuração

```
16  fs.createReadStream("./players_stats.csv")
17    .pipe(parse({ delimiter: ",", from_line: 1 }))
18    .on("data", (row) => {
19      if (line == 1) {
20        columnNames = row;
21        for (let i = 1; i < columnNames.length; i++) {
22          maxValues.push(null);
23          minValues.push(null);
24        }
25      } else {
26        for (let i = 1; i < columnNames.length; i++) {
27          const value = parseFloat(row[i]);
28          if (!maxValues[i] || value > maxValues[i]) {
29            maxValues[i] = value;
30          }      You, 24 hours ago • initial commit …
31          if (!minValues[i] || value < minValues[i]) {
32            minValues[i] = value;
33          }
34        }
35      }
36      line++;
37    })
```

Fonte: autores

tados e os disponibiliza em gráficos para visualização de dados.

Na Figura 27, é possível ver três métodos desenvolvidos nesse código, um capaz de gerar uma regressão linear; um capaz de gerar um gráfico de barras, e um capaz de gerar um histograma.

5.5 Iterações de Desenvolvimento

Essa seção diz respeito à documentação das principais etapas do desenvolvimento, realizada ao longo do Trabalho de Conclusão de Curso 2, orientando-se pela metodologia descrita na Seção 4.4.

5.5.1 *Sprints* Planejadas

O desenvolvimento da aplicação foi planejada tendo em mente um período de 8 semanas para a contrução da *Multiagent Matchmaking API*. Para isso, foram utilizadas *sprints* com uma semana de duração, totalizando oito *sprints*.

Figura 26 – Código de Envio de Jogadores

```
amqp.connect("amqp://user:password@localhost:5672", (error, connection) => {
  if (error) console.log(error);

  connection.createChannel((error, channel) => {
    if (error) console.log(error);

    channel.assertQueue("INCOMING_QUEUE", { durable: true });

    players.forEach((player) => {
      channel.publish(
        "DX_PLAYER",
        "INCOMING_QUEUE",
        Buffer.from(JSON.stringify(player)),
        (error) => {
          if (error) throw error;
        }
      );
    });

    channel.close();
  });
});
```

Fonte: autores

Figura 27 – Código de Análise de Dados

```
def linear_regression(xAxis, yAxis):
    sns.regplot(x=xAxis, y=yAxis, line_kws={"color": "r", "alpha": 0.7, "lw": 5 })
✓ 0.2s


def bar_chart(title_name, xAxis_name, yAxis_name, xAxis_values, yAxis_values):
    plt.bar(xAxis_values,yAxis_values)
    plt.title(title_name)
    plt.xlabel(xAxis_name)
    plt.ylabel(yAxis_name)
    plt.show()
✓ 0.2s


def generate_hist(title, values):
    plt.title(title)
    plt.hist(values)
    plt.show()
```

Fonte: autores

Seguem os requisitos planejados, conforme a Seção 5.2, e que foram desenvolvidos em cada uma das *sprints*:

- ***Sprint* 01**: FE01, FE06 e FE08;
- ***Sprint* 02**: FE07, FE11 e FE14;

- ***Sprint* 03**: FE03 e FE09;
- ***Sprint* 04**: FE15 e FE16;
- ***Sprint* 05**: FE04 e FE12;
- ***Sprint* 06**: FE13 e FE17;
- ***Sprint* 07**: essa *sprint* foi planejada para ajustes de código e realização de tarefas atrasadas, e
- ***Sprint* 08**: essa *sprint* foi planejada para ajustes de código e realização de tarefas atrasadas.

5.6 Resumo do Capítulo

Nesse capítulo, a *Multiagent Matchmaking API* foi apresentada e detalhada. Primeiramente, foram retomados aspectos de Contextualização, os quais motivaram a realização desse trabalho. Depois, foram acordados os Requisitos da Aplicação, os quais compreendem as principais funcionalidades elicitadas, priorizadas e implementadas na *Multiagents Matchmaking API*. Em seguida, foi apresentada a Arquitetura da API, com foco nos principais componentes arquiteturais e detalhamentos lógicos, além do funcionamento e da comunicação entre os agentes, ilustrando cada aspecto com modelos e trechos de código.

Consta ainda um conjunto de *Scripts* Auxiliares, utilizados durante o teste e a análise de resultados, conferindo, adicionalmente, facilitadores para continuidade desse projeto, pelos autores ou futuros interessados. Por fim, foram, brevemente, apresentadas as Iterações compreendidas no desenvolvimento da API, orientando-se pela metodologia de desenvolvimento definida na Seção 4.4, e com foco nas *Sprints* e no *Backlog* do Produto.

6 Análise de Resultados

Esse capítulo é responsável por descrever e analisar os resultados obtidos ao longo do desenvolvimento do projeto. Serão apresentadas as tarefas elaboradas durante a fase Pesquisa-Ação, considerando o Primeiro Ciclo da Pesquisa-Ação buscando compreender a Problemática e realizando a Coleta de Dados; o Segundo Ciclo da Pesquisa-Ação com a Elaboração do Plano de Ação, e a Divulgação de Resultados. Por fim, tem-se o Resumo do Capítulo.

6.1 Fases da Pesquisa-Ação

Como descrito no capítulo de Metodologia, esse trabalho utiliza da pesquisa-ação para poder realizar a análise de resultados. A pesquisa teve presença ativa dos pesquisadores, desde a fase de planejamento da pesquisa até as suas partes finais de análise de dados. Em um primeiro momento, foram realizada as Coletas de Dados referentes à *Multiagent Matchmaking API*. Logo em seguida, foram realizadas análises e interpretações sobre os dados levantados, afim de compreender se o Sistemas Multiagentes são adequados para o problema de Sistemas de *Matchmaking*. Por fim, um Plano de Ação foi elaborado com possíveis melhorias sobre o desenvolvimento do projeto, seguido da Divulgação de Resultados e da conclusão dos autores sobre o projeto.

6.2 Primeiro Ciclo - Compreendendo o Problema

Seguindo o processo iterativo de ciclos dentro da **Pesquisa-Ação**, o primeiro ciclo foi estabelecido a partir de uma análise da bibliografia de estudos sobre outros Sistemas *Matchmaking*. Foram levantados critérios de análise a partir dessas referências, iniciando, assim, um levantamento de dados da aplicação. Para realizar a coleta de dados, foram estudadas duas bases de dados de diferentes jogos.

6.2.1 Problemática

Para compreender a problemática, foram analisados, com base em artigos, os principais aspectos apontados como problemas referentes aos Sistemas *Matchmaking*. Esses aspectos foram analisados e levados em consideração para entender se o uso de Sistemas Multiagentes, aplicado à problemática de Sistema *Matchmaking*, é ou não adequado.

Dentre os principais aspectos analisados, cabe discorrer sobre aspectos já levantados no Referencial Teórico, abordados por VÉRON; MARIN; MONNET (2014), sobre o

impacto de Sistemas *Matchmaking* na experiência dos jogadores, cabendo considerar:

- **Tempo de espera do jogador dentro da fila do *Matchmaking* até a formação de uma partida:** Esse aspecto foi analisado levando em consideração a colocação dos dados do jogador na fila do RabbitMQ até a sua inserção em um *lobby* de jogo;
- **Precisão da formação da partida entre os jogadores:** Esse aspecto foi analisado com base na comunicação entre os agentes, e nos algoritmos que calculam desvio padrão de categorias de habilidades dos jogadores, e
- **Latência do tempo de resposta do servidor entre os jogadores:** Esse aspecto foi descartado em um primeiro momento, tendo em vista que as bases de dados utilizadas para a experimentação não continham informação sobre localização dos jogadores. Entretanto, recomenda-se o tratamento desse aspecto em trabalhos futuros, buscando tratar de forma mais adequada a regionalidade, por exemplo, dos jogadores.

Ambos os aspectos testados, **Tempo de espera do jogador e Precisão da formação da partida**, foram testados com quantidades de agentes diferentes, afim de compreender se isso impactaria no resultado das amostras.

6.2.2 Coleta de Dados

Para a Coleta de dados, foi realizada uma integração com o MongoDB para cumprimento do registro dos experimentos. Anteriormente, foram levantadas duas bases de dados da plataforma KAGGLE. Essas bases foram de jogos diferentes, com o intuito de compreender se o Sistema atenderia diferentes tipos de jogos. As bases levantadas foram:

- ***Top Women Chess Players*:** Uma base de dados com informação sobre classificação de jogadoras de Xadrez do mundo inteiro (OJHA, 2020). Essa base apresenta dados a respeito de 8532 jogadoras diferentes e 10 colunas de dados, sendo que apenas 3 delas podem ser utilizadas como critérios para avaliar a habilidade das jogadoras, e
- ***Fortnite Players Stats*:** Uma base de dados com informações sobre classificação de jogadores do jogo FORTNITE (ALI, 2021). Essa base apresenta dados a respeito de 1435 jogadores diferentes e 37 colunas de informação, sendo que dessas 37 colunas conseguimos extrair dados a respeito de 5 modalidades de jogo diferentes, cada uma das modalidades apresenta 7 colunas de dados que podem ser utilizadas como critérios para avaliar a habilidade dos jogadores.

As bases de dados levantadas foram tratadas para um modelo de *json*, Figura 28 e Figura 29, e foram inseridas diretamente na fila, como se um jogador estivesse solicitando a entrada em uma partida.

Figura 28 – Exemplo do dado do jogador da base *Top Women Chess Players*


```
{
    "playerId": "Hou, Yifan",
    "playerData": [
        {
            "key": "Standard_Rating",
            "value": 2658
        },
        {
            "key": "Rapid_rating",
            "value": 2621
        },
        {
            "key": "Blitz_rating",
            "value": 2601
        }
    ]
},
```


Fonte: autores

Figura 29 – Exemplo do dado do jogador da base *Fortnite Players Stats*


```
{
    "playerId": "Prospering",
    "playerData": [
        { "key": "Solo score", "value": 2476763 },
        { "key": "Solo top1", "value": 1828 },
        { "key": "Solo kd", "value": 4.37 },
        { "key": "Solo winRatio", "value": 18 },
        { "key": "Solo matches", "value": 10150 },
        { "key": "Solo kills", "value": 36328 },
        { "key": "Solo minutesPlayed", "value": 81389 },
        { "key": "Duos score", "value": 4702684 },
        { "key": "Duos top1", "value": 5584 },
        { "key": "Duos kd", "value": 10.71 },
        { "key": "Duos winRatio", "value": 45.7 },
        { "key": "Duos matches", "value": 12229 },
        { "key": "Duos kills", "value": 71137 },
        { "key": "Duos minutesPlayed", "value": 133725 },
        { "key": "Trios score", "value": 299128 },
        { "key": "Trios top1", "value": 244 },
        { "key": "Trios kd", "value": 6.65 },
        { "key": "Trios winRatio", "value": 31.2 },
        { "key": "Trios matches", "value": 783 },
        { "key": "Trios kills", "value": 3584 },
        { "key": "Trios minutesPlayed", "value": 10280 },
        { "key": "Squads score", "value": 3640415 },
        { "key": "Squads top1", "value": 5116 },
        { "key": "Squads kd", "value": 6.88 },
        { "key": "Squads winRatio", "value": 31.7 },
        { "key": "Squads matches", "value": 16131 },
        { "key": "Squads kills", "value": 75787 },
        { "key": "Squads minutesPlayed", "value": 117967 },
        { "key": "LTM score", "value": 101589 },
        { "key": "LTM top1", "value": 170 },
        { "key": "LTM top3", "value": 118 },
        { "key": "LTM kd", "value": 5.78 },
        { "key": "LTM winRatio", "value": 26.8 },
        { "key": "LTM matches", "value": 634 },
        { "key": "LTM kills", "value": 2682 },
        { "key": "LTM minutesPlayed", "value": 2865 }
    ]
}
```


Fonte: autores

Os dados que foram inseridos na aplicação e, posteriormente, processados, foram armazenados e geraram dados sobre o *lobby* que foi criado. Na Figura 30, é possível ver

uma amostra desses dados.

Figura 30 – Exemplo do dado gerado sobre um *Lobby*

Fonte: autores

6.3 Segundo Ciclo - Tratamento dos Dados

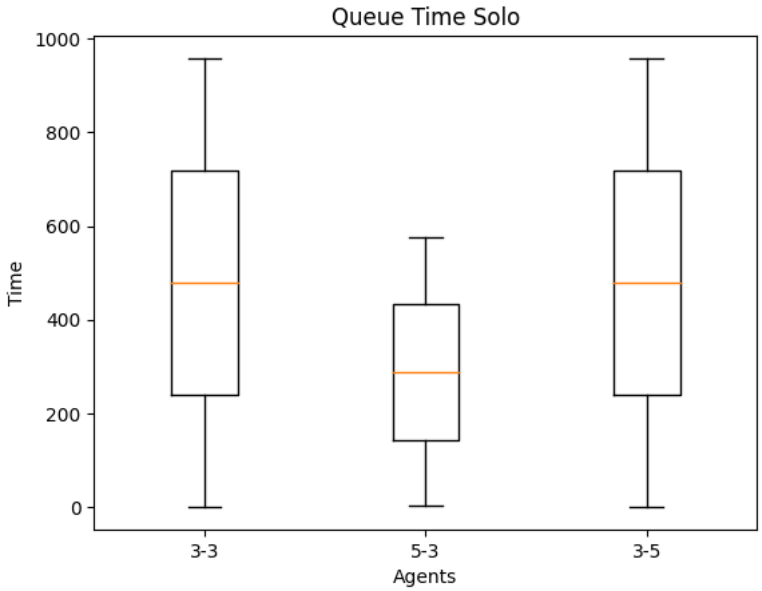
Nessa etapa, os dados coletados da **API** foram tratados e convertidos para o formato de gráficos, no intuito de facilitar a visualização bem como a extração de informações. Os dados levantados da *API* foram dados referentes ao seu uso, bem como à formação de partida e à jornada do jogador ao longo da *Multiagent Matchmaking API*.

Os dados analisados foram feitos para diferentes categorias na base do ALI (2021), levando em consideração partidas que são *solo*, apenas um jogador contra o restante, e partidas que são em *squad*, jogadores formados em times de até 4 pessoas. Já para a base do OJHA (2020), só foram realizadas análises sobre os dados de partidas simples entre os jogadores, e levados em consideração apenas dados referentes ao *match* dos jogadores. Todos os dados foram analisados considerando diferentes configurações da organização dos Agentes, sendo analisadas, principalmente, as organizações com: (i) 3 Agentes do tipo *Adder* e 3 Agentes do tipo *Lobby Organizer*; (ii) 5 Agentes do tipo *Adder* e 3 Agentes do tipo *Lobby Organizer*, e (iii) 3 Agentes do tipo *Adder* e 5 Agentes do tipo *Lobby Organizer*.

Inicialmente, foram analisados dados referentes à espera do jogador na fila, visando ser inserido no *lobby*. As Figura 31 e 32 exibem três Diagramas de Caixa com a análise de dados sobre a base do ***Fortnite*** (ALI, 2021), com o tempo de espera dos jogadores na fila para entrar em uma partida, formatados para diferentes configurações de Agentes.

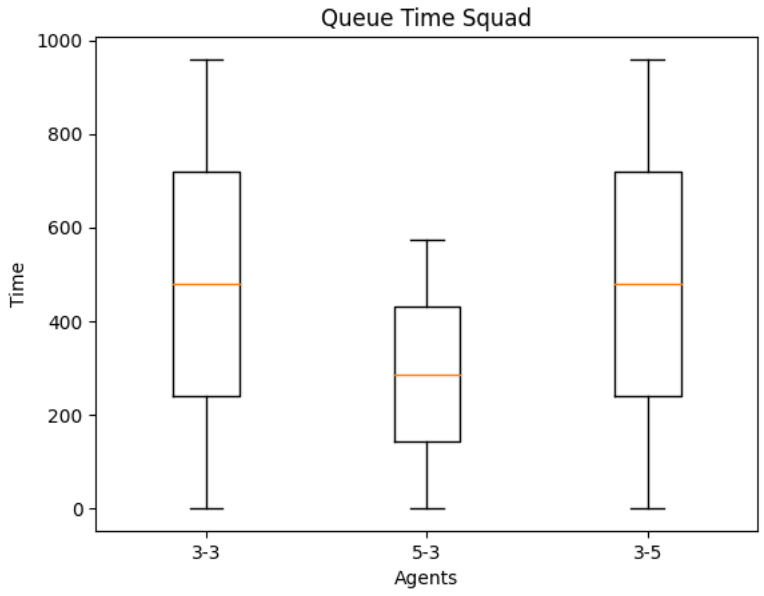
Tanto para a Figura 31, quanto para a Figura 32, é possível observar que no Sis-

Figura 31 – Diagrama de Caixa do Tempo do jogador na Fila para partida *Solo* do *Fortnite*.



Fonte: autores

Figura 32 – Diagrama de Caixa do Tempo do jogador na Fila para partida de *Squad* do *Fortnite*.



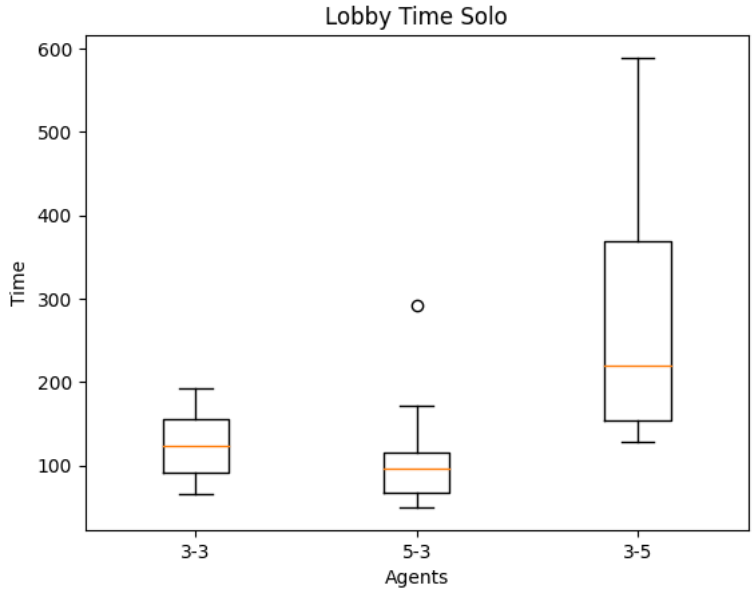
Fonte: autores

tema, quando configurado com 5 Agentes do tipo *Adder*, o tempo de espera dos jogadores acaba sendo menor na fila. Isso ocorre pelo fato dos Agentes do tipo *Adder* serem os principais responsáveis por consumirem os dados das filas, ou seja, quanto mais *Adder Agents*, maior a quantidade de consumidores da fila de entrada de jogadores. Apesar desse tempo menor, os tempos de espera de alguns jogadores estão discrepantes, chegando próximo dos 1000 segundos. Como a estrutura de inserção de jogadores na *API* é via uma fila, e nos testes constam inseridos vários jogadores de uma só vez, os jogadores que ficaram ao

final fila acabaram incorrendo em um grande tempo de espera.

Além do tempo de espera dos jogadores na fila, foram analisados o tempo de construção das partidas. O tempo de construção das partidas é importante para entender parte do comportamento dos Agentes do tipo *LobbyOrganizer*. As Figuras 33 e 34 exibem três Diagramas de Caixa com a análise de dados sobre a base do ***Fortnite*** (ALI, 2021), com o tempo de organização de uma partida formatado para diferentes configurações de Agentes.

Figura 33 – Diagrama de Caixa do Tempo de formação do *Lobby* para partida *Solo* do *Fortnite*



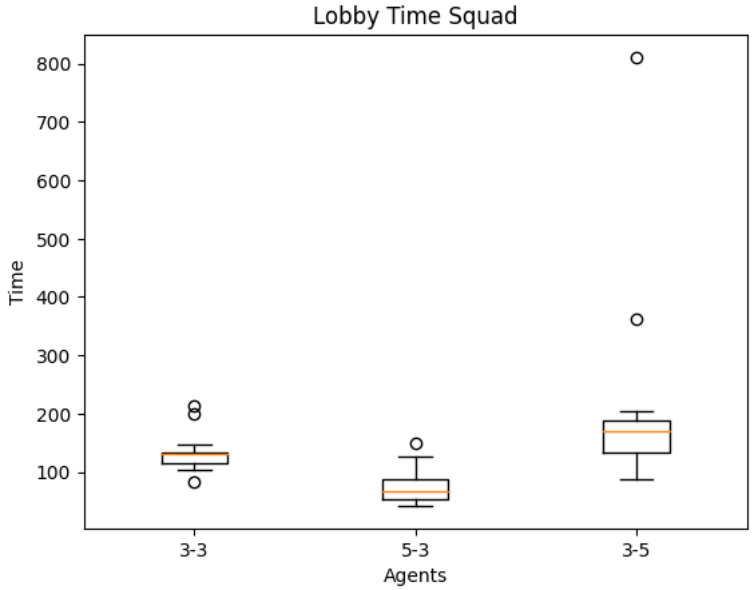
Fonte: autores

Para os gráficos de tempo de formação dos *lobbies*, é possível observar que alguns dados ficaram discrepantes em comparação aos limites inferiores e superiores dos Diagramas de Caixa. Os tempos de formação dos *lobbies* são, consideravelmente, menores em comparação aos tempos de espera dos jogadores nas filas. Isso decorre dos Agentes do tipo *LobbyOrganizer* não dependerem de uma fila para realizarem os seus cálculos, mas apenas dos Agentes do tipo *Adder*, os quais enviam jogadores que são aceitáveis para a construção do *lobby*.

Além dos tempos de formação das partidas, foi analisada a precisão de formação das partidas. Essa precisão leva em consideração o desvio absoluto médio entre os jogadores. O cálculo do desvio absoluto médio foi feito com base no cálculo do desvio padrão entre os jogadores nas partidas que foram construídas. Nas Figuras 35, 37, 36, 38, 39 e 40, são ilustrados os resultados desse desvio.

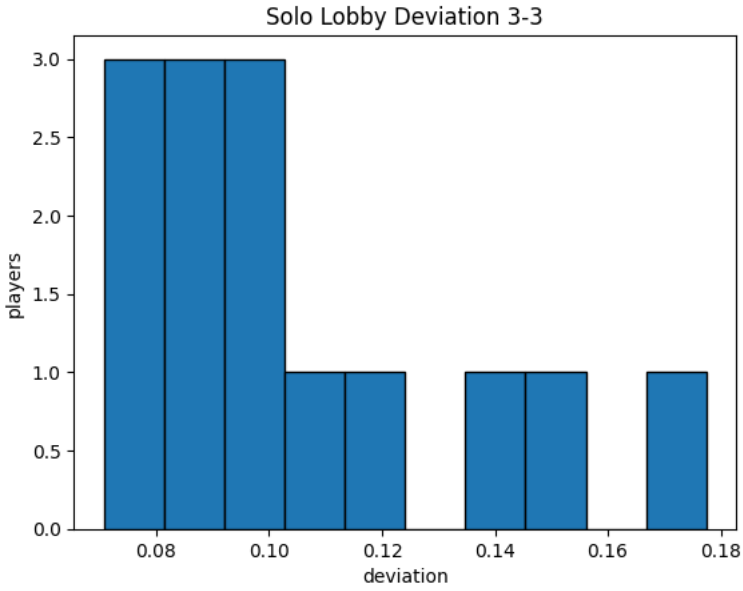
Para as partidas *solos* no *Fortnite*, foram observadas que a diferente formação entre tipos de Agentes teve impacto em seus resultados. Na configuração de 3 Agentes *Adder* e 3 Agentes *LobbyOrganizer* e na configuração 3 Agentes *Adder* e 5 Agentes *LobbyOrganizer*,

Figura 34 – Diagrama de Caixa do Tempo de formação do *Lobby* para partida *Squad* do *Fortnite*.



Fonte: autores

Figura 35 – Histograma do Desvio Absoluto Médio 3 *Adder* e 3 *LobbyOrganizer* para *Solo*

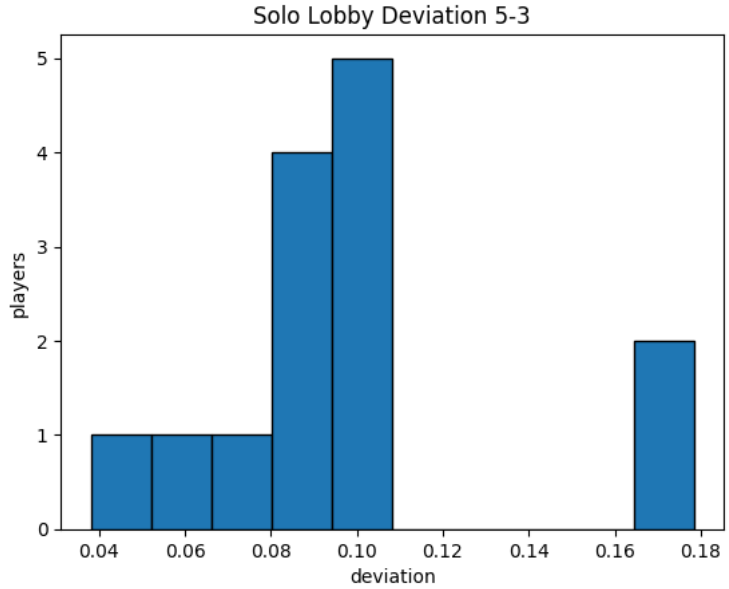


Fonte: autores

Figura 35, Figura 37 e Figura 36, é possível observar que existem agrupamentos de partidas que ficaram mais à direita dos gráficos, formando pequenos grupos, e deixando algumas partidas mais isoladas.

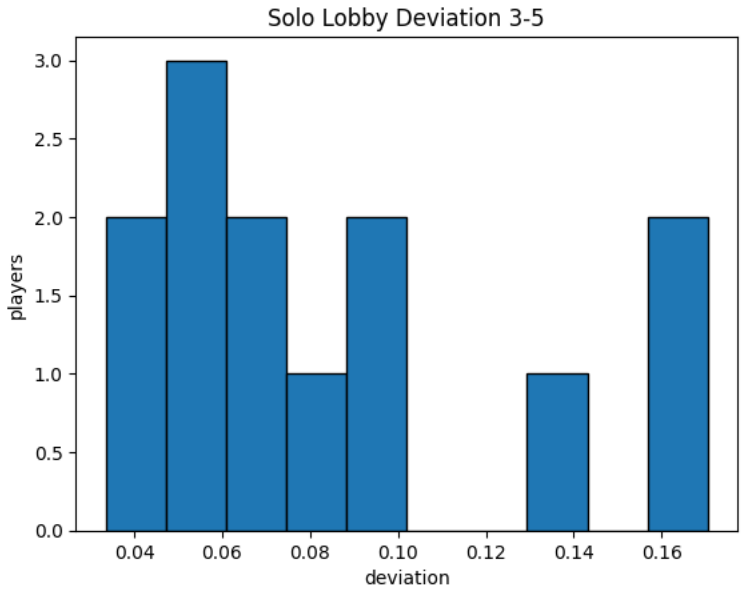
Apenas para as partidas em *Squad*, os dados ficam mais precisos, ou seja, mais próximos uns dos outros. Os desvios absolutos médios desses *lobbies* acabam revelando comportamentos discrepantes uns dos outros, como é possível observar nas Figuras 38, 40 e 39. Isso ocorre pois alguns dos *lobbies* que foram construídos com os jogadores que

Figura 36 – Histograma do Desvio Absoluto Médio 5 *Adder* e 3 *LobbyOrganizer* para *Solo*



Fonte: autores

Figura 37 – Histograma do Desvio Absoluto Médio 3 *Adder* e 5 *LobbyOrganizer* para *Solo*

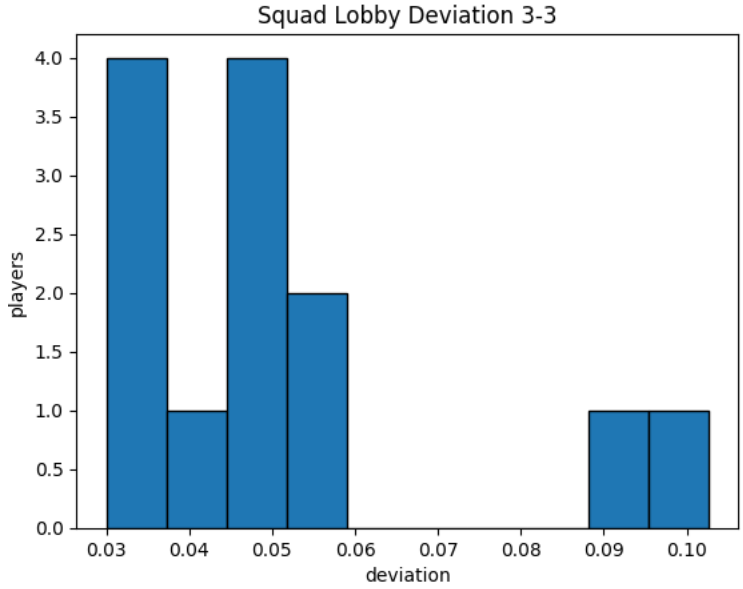


Fonte: autores

sobraram, tinham o seu desvio absoluto médio muito alto em comparação com o restante. A alta do valor de desvio absoluto médio, ocorre devido aos jogadores que sobram na fila no processamento do *matchmaking*, para não ficarem esperando muito tempo na fila, eles são inseridos em partidas com outros jogadores que podem ter uma diferença de valor de habilidade alta entre eles.

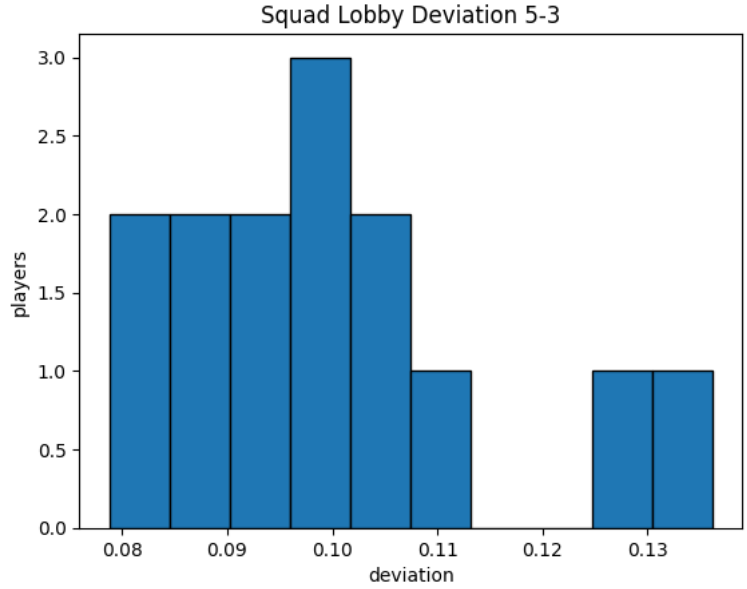
Por fim, analisando a base de dados do OJHA (2020), os dados ficaram praticamente todos acumulados em apenas uma barra. Isso acontece por conta da base de dados possuir informações sobre as jogadoras muito próximas umas das outras em classificação.

Figura 38 – Histograma do Desvio Absoluto Médio 3 *Adder* e 3 *LobbyOrganizer* para *Squad*



Fonte: autores

Figura 39 – Histograma do Desvio Absoluto Médio 5 *Adder* e 3 *LobbyOrganizer* para *Squad*



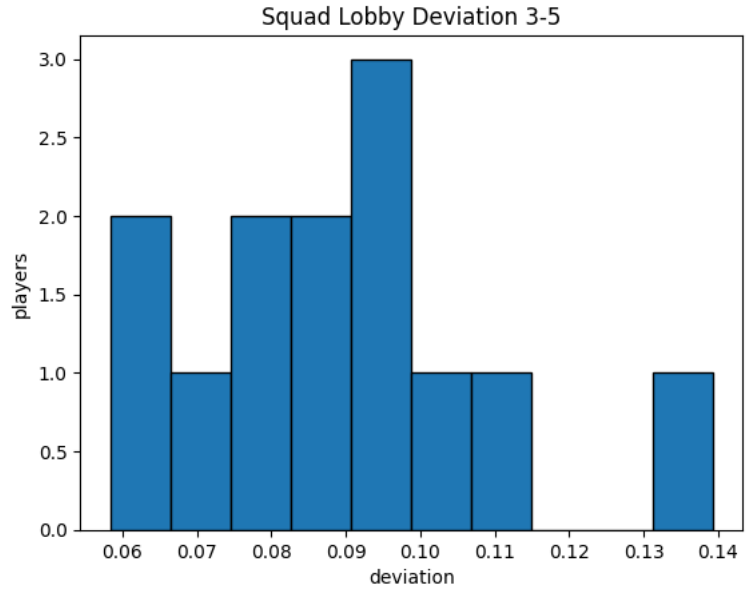
Fonte: autores

Tais informações estão ilustradas nas Figuras 41, 42 e 43.

6.3.1 Plano de Ação

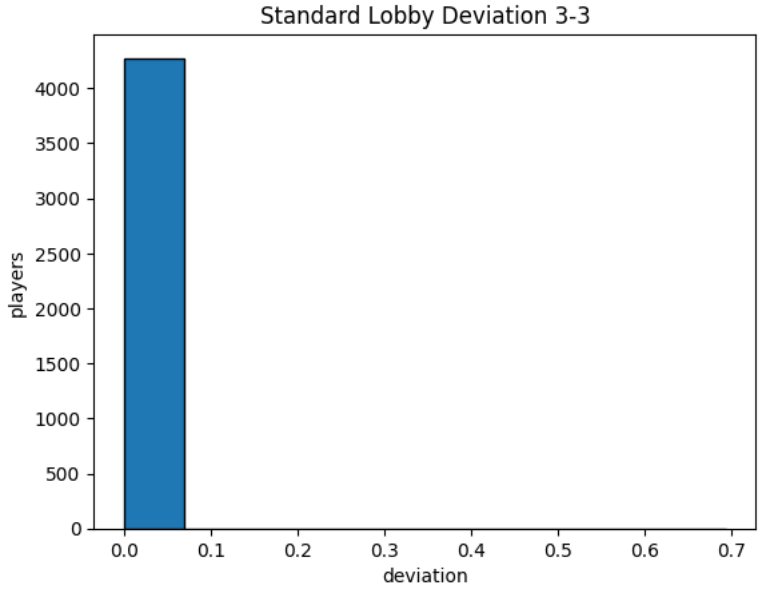
A partir dos dados coletados, junto com as análises realizadas sobre esses dados, foram identificados pontos de melhorias na *API*, sendo elaborado um Plano de Ação. O Plano de Ação torna claro as ações que devem ser realizadas para que o projeto possa

Figura 40 – Histograma do Desvio Absoluto Médio 3 *Adder* e 5 *LobbyOrganizer* para *Squad*



Fonte: autores

Figura 41 – Histograma do Desvio Absoluto Médio 3 *Adder* e 3 *LobbyOrganizer* para a Base de Xadrez

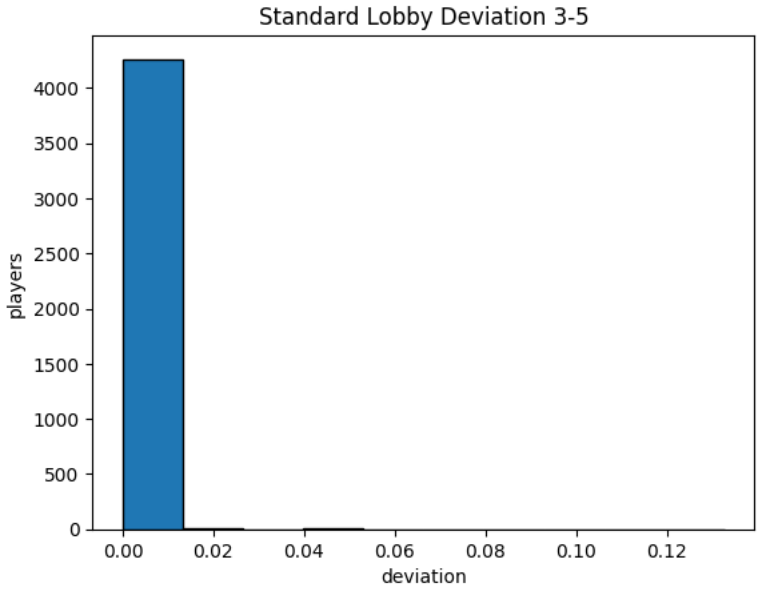


Fonte: autores

evoluir.

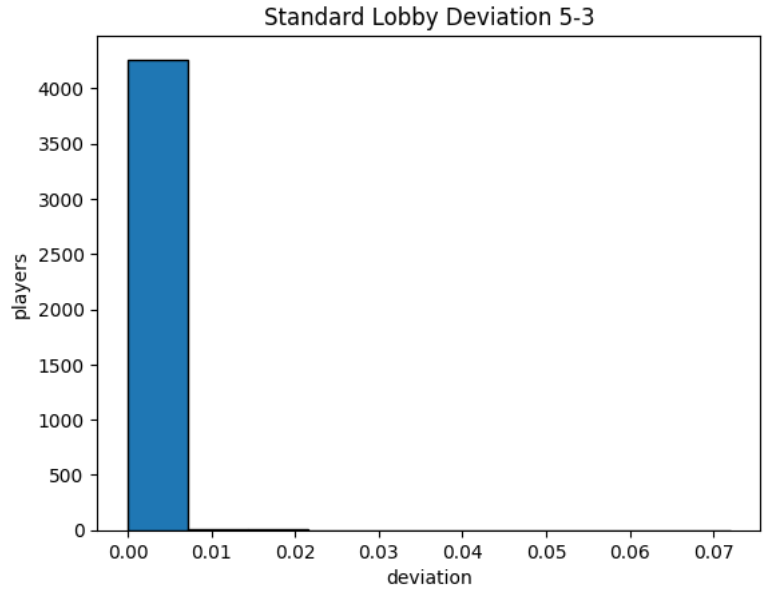
- Trocar o comportamento dos Agentes do tipo *Adder* para um comportamento cíclico, para que possam consumir os jogadores da fila de forma mais rápida, e
- Trocar o coeficiente de *score* máximo, para melhorar a dispersão dos desvios absolutos médios nos *Lobbies*.

Figura 42 – Histograma do Desvio Absoluto Médio 3 *Adder* e 5 *LobbyOrganizer* para a Base de Xadrez



Fonte: autores

Figura 43 – Histograma do Desvio Absoluto Médio 5 *Adder* e 3 *LobbyOrganizer* para a Base de Xadrez



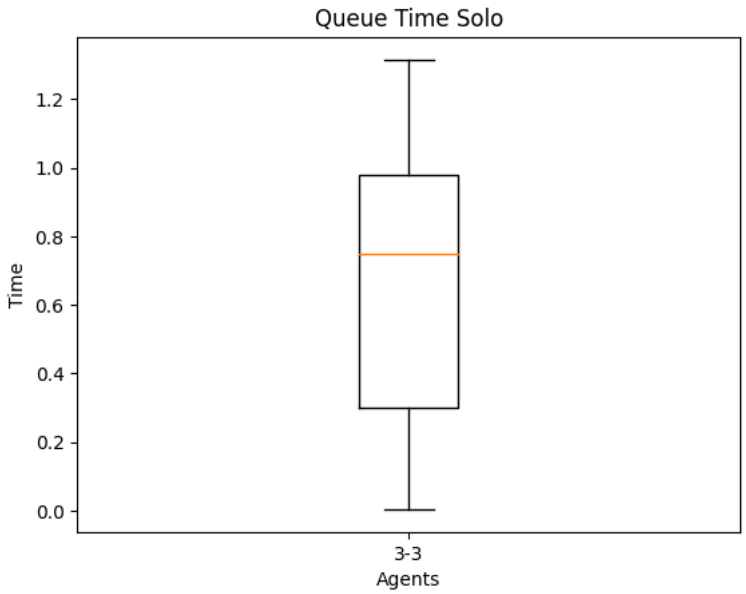
Fonte: autores

6.3.2 Divulgação de Resultados

Conforme colocado anteriormente, com o Plano de Ação, buscou-se realizar ações no intuito de melhorar a *Multiagent Matchmaking API*. As ações levaram em consideração duas premissas. A seguir, segue cada premissa e o respectivo resultado alcançado:

- Trocar o comportamento dos Agentes do tipo *Adder* para um comportamento cíclico visando consumir os jogadores da fila de forma mais rápida.

Figura 44 – Diagrama de Caixa do tempo para a base do *Fortnite*.



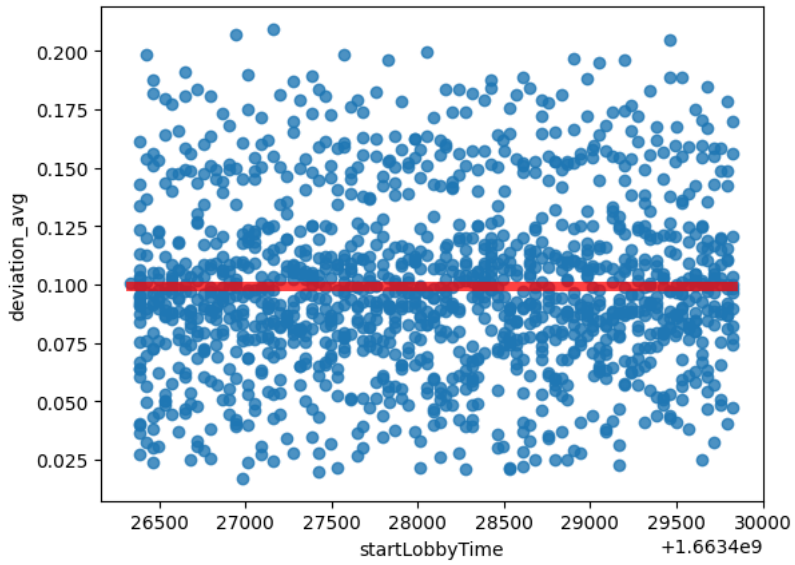
Fonte: autores

Resultados: A mudança para o comportamento cíclico causou uma mudança significativa no tempo de espera do jogador na fila. Como o comportamento dos Agentes do tipo *Adder* tornaram-se cíclicos, o tempo de espera do jogador na fila reduziu-se significativamente, como pode ser visto no Diagrama de Caixa da Figura 44. Dentre as razões pelas quais isso ocorre, destaca-se o fato do comportamento cíclico ficar em modo de espera constante, aguardando o recebimento do evento esperado. Uma vez que esse evento chega, imediatamente, o comportamento é disparado, não demandando chamada de um método ou comportamento para execução da ação desejada. Há, nesse contexto, redução de tempo, e

- Trocar o coeficiente de *score* máximo, para melhorar a dispersão dos desvios absolutos médios dentro dos *Lobbies*.
  Resultados: a *Multiagent Matchmaking API* utiliza um coeficiente de *score* máximo para determinar o maior valor aceitado pelo agente *Adder* na negociação com os agentes *Lobby Organizers*. Havia uma premissa de que a diminuição do valor desse coeficiente melhoraria a dispersão dos desvios absolutos médios de cada *lobby*. Considerando essa premissa, diminui-se o valor desse coeficiente, que originalmente era 0.1, para 0.06. Porém, os dados obtidos através desse teste foram muito similares aos resultados obtidos com o valor de coeficiente anterior. Por conta disso, foi desconsiderada a premissa de que essa variável poderia melhorar a dispersão dos desvios absolutos médios.
  Concluiu-se então que para a melhoria da dispersão dos desvios absolutos médios nos *Lobbies* ocorrer, seria necessário repensar o método utilizado para a atribuição dos *scores*. Por conta disso, foi definido pelos autores que o atendimento desse as-

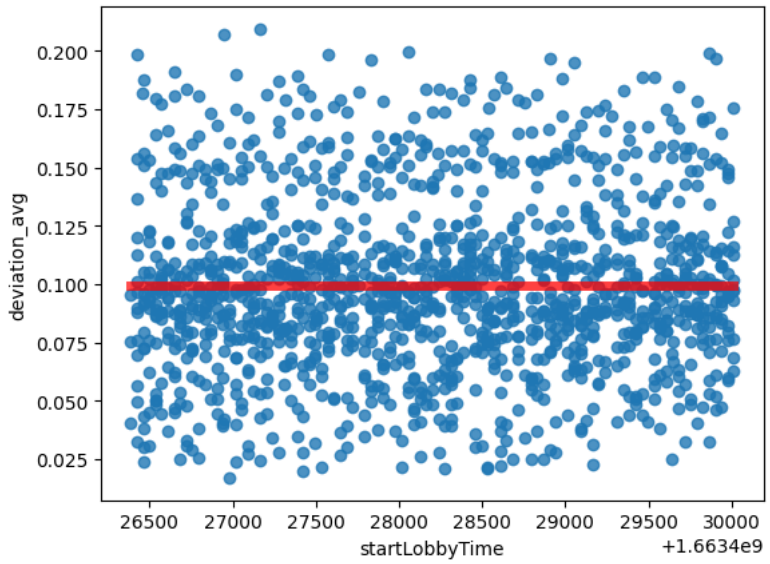
pecto não estaria no escopo desse Trabalho de Conclusão de Curso. As similaridades entre os resultados, tomando como bases o coeficiente 0.1 e o coeficiente 0.06, são ilustradas nas Figuras 45 e 46.

Figura 45 – Regressão Linear do Desvio Absoluto Médio com coeficiente 0.1 base do *Fortnite*



Fonte: autores

Figura 46 – Regressão Linear do Desvio Absoluto Médio com coeficiente 0.06 base do *Fortnite*



Fonte: autores

6.3.2.1 Inferências Qualitativas

Após toda análise dos dados coletados da API, pode-se inferir informações qualitativas sobre os problemas analisados. A partir do estudo feito sobre a mudança de

comportamento do Agente tipo *Adder* de um comportamento baseado em tempo para um comportamento cíclico, observou-se que houve uma mudança considerável com relação ao tempo de espera do jogador na fila, ou seja, os jogadores passaram menos tempo aguardando na fila à espera de uma partida. Apoiado nos trabalhos de XU; YU; WU (2021) e VÉRON; MARIN; MONNET (2014), essa mudança torna-se positiva, tendo em vista que alguns jogos como o LEGENDS (2022), podem levar até 45 minutos para a formação de uma partida, atrapalhando diretamente a experiência do jogador.

Já a análise qualitativa sobre a precisão da formação da uma partida, essa torna-se mais complicado inferir alguma informação qualitativa. Apesar de muitas partidas possuírem um desvio absoluto médio baixo, criando um pareamento entre jogadores mais equilibrado, a informação de que a partida está mais justa ou não, só seria confirmada com a integração da *Multiagent Matchmaking API* com algum jogo de fato, e experimentado com jogadores reais. Essa conclusão foi alcançada, principalmente, ao analisar trabalhos de outros autores como por exemplo de CLAYPOOL M. (2015), onde em sua conclusão observa que a opinião dos jogadores sobre o equilíbrio das partidas é fortemente influenciada pelo fato deles terem vencido ou perdido uma partida.

6.4 Resumo do Capítulo

Esse Capítulo focou na análise de resultados da *Multiagent Matchmaking API*. Inicialmente, foram descritas as Fases da Pesquisa-Ação. Tomando como base o processo iterativo de ciclos da **Pesquisa-Ação**, foram relatados sobre os acontecimentos relacionados ao Primeiro Ciclo e ao Segundo Ciclo. Em cada um desses ciclos, foram descritos dados coletados e informações obtidas através desses dados.

Diante do exposto, percebe-se que o uso de multiagentes, nesse estudo exploratório em Sistemas de *Matchmaking*, mostrou-se interessante e pertinente, com destaque para:

- **PONTO DE REFLEXÃO 01**: é possível provocar reduções significativas no tempo de espera dos jogadores na fila, simplesmente, aumentando o número de Agentes do tipo *Adder*. Isso é, particularmente, demonstrado no capítulo, ao aumentar o número de Agentes do tipo *Adder* de 3 para 5;
- **PONTO DE REFLEXÃO 02**: é possível perceber que os tempos de formação dos *lobbies* são, consideravelmente, menores, se comparados aos tempos de espera dos jogadores nas filas. Dessa forma, os tempos de construção das partidas em si, tarefa essa desempenhada pelos *Lobby Organizers*, não são tão significativos na solução apresentada. O gargalo, portanto, é no tempo de espera dos jogadores na fila, o qual é facilmente tratado, em multiagentes, aumentando o número de Agentes do tipo

Adder, conforme o **PONTO DE REFLEXÃO 01**. Isso se deve à escalabilidade facilitada, sendo essa uma característica intrínseca de Sistemas Multiagentes;

- **PONTO DE REFLEXÃO 03**: é possível notar que, ao se usar recursos mais sofisticados da Plataforma JADE, tal como comportamentos mais especializados (ex. comportamento cíclico), pode-se ter ganhos na solução. Isso ocorreu, ao se optar pela implementação do comportamento cíclico para o caso dos Agentes do tipo *Adder*, usufruindo do fato desse comportamento ser programado, internamente na Plataforma JADE, com padrões de projeto e orientação a eventos, permitindo reações comportamentais bem mais rápidas, e melhorando o tempo de consumo dos agentes na fila. Portanto, reduzindo o tempo de espera dos jogadores na fila, e
- **PONTO DE REFLEXÃO 04**: outro ponto, não muito evidenciado nos resultados expostos, mas também observado pelos autores ao longo do estudo exploratório, foi a maior possibilidade de lidar com tolerância a falhas. Nesse contexto, deve-se, primeiramente, considerar a facilidade de escalabilidade, podendo usar vários agentes em atendimento às diferentes demandas, inerentes à solução, tais como: consumo de agentes esperando na fila e construção das partidas em si. Em segundo, deve-se considerar que esses agentes são autônomos, e atuam de forma assíncrona, em obediência aos princípios de Sistemas Multiagentes. Por fim, quando um problema ocorre, ao ponto de inviabilizar a atuação de um dado agente, em uma dada posição, o sistema pode continuar rodando normalmente, considerando os demais agentes. Sendo assim, o sistema não deixará de funcionar, devido à não atuação de um ou outro agente, uma vez que outros agentes passarão a assumir as ações pendentes.

7 Conclusão

Sistemas *Matchmaking* de jogos são, de certa forma, polêmicos no meio dos jogadores. Muitos jogos acabam tendo muitas dificuldades em conciliarem os seus sistemas de pareamentos de jogadores de uma forma que satisfaça a sua comunidade, conseguindo diminuir os impactos negativos sobre a experiência do jogo. A presente pesquisa não teve como objetivo mitigar todos os problemas relacionados a um Sistema *Matchmaking*, mas apenas estudar e explorar uma nova abordagem, utilizando-se do paradigma de Sistemas Multiagentes.

O paradigma de Sistemas Multiagentes possui suas particularidades que podem tornar boa parte da sua compreensão um pouco difícil. Entretanto, Sistemas Multiagentes podem fornecer recursos para auxiliar em problemas tipicamente encontrados nos Sistemas de *Matchmaking* atuais, como o desbalanceamento de habilidades entre jogadores em uma mesma partida. Problema esse que gera frustração, e cria um impacto negativo sobre a experiência de jogo. A cognição de um Agente de Software e o seu trabalho cooperativo com outros Agentes podem tornar todo o processo de pareamento de jogadores mais vantajoso. O trabalho colaborativo dos agentes faz com que o processo de decisão para parear jogadores possa ser mais justo e mais eficiente que os sistemas atuais. Essas premissas, com os resultados obtidos nesse trabalho, foram esclarecidas e confirmadas.

Esse capítulo, portanto, procura conferir um fechamento à escrita do projeto, resgatando os objetivos, Geral e Específicos, e reportando seus status; bem como conferindo resposta à questão de pequisa. Por fim, são apresentadas ideias para trabalhos futuros.

7.1 Status do Trabalho

Com base no cronograma apresentado na Tabela 4, pode-se avaliar o *status* das atividades correspondentes ao TCC1. Essa análise pode ser vista na Tabela 6.

Com base no cronograma apresentado na Tabela 5, pode-se avaliar o *status* das atividades correspondentes ao TCC2. Essa análise pode ser vista na Tabela 7.

Além das atividades realizadas, também é valido apresentar o *status* do trabalho em relação aos objetivos específicos, que foram propostos na Seção 1.4. Pode-se ver os *status* dos objetivos na Tabela 8.

7.2 Objetivos Concluídos

Retomando os objetivos específicos apresentados no Capítulo de 1, descritos como:

Tabela 6 – Status das Atividades TCC1

Atividades	***Status***
Definir Tema	Concluída
Levantamento Bibliográfico	Concluída
Elaborar Introdução	Concluída
Definir Referencial Teórico	Concluída
Definir Metodologia	Concluída
Definir Suporte Tecnológico	Concluída
Refinar Escopo do Trabalho	Concluída
Elicitar Requisitos	Concluída
Definir Proposta	Concluída
Refinar Monografia	Concluída
Apresentação à Banca	Concluída

Fonte: autores

Tabela 7 – Status das Atividades TCC2

Atividades	***Status***
Aplicar Correções da Banca	Concluída
Desenvolvimento da API	Concluída
Análise de Resultados	Concluída
Documentar Resultados	Concluída
Refinar Monografia	Concluída
Apresentação à Banca	Concluída

Fonte: autores

Tabela 8 – Status dos Objetivos Específicos

Objetivos Específicos	***Status***
Investigar Sistemas Multiagentes, em termos de literatura especializada	Concluído
Investigar Sistemas *Matchmaking*, em termos de literatura especializada	Concluído
Especificar, Projetar, Desenvolver e Documentar uma API, orientada a SMA, capaz de auxiliar na criação de algoritmos de *Matchmaking*	Concluído
Analisar os Resultados Obtidos, procurando documentar as impressões percebidas, com foco na questão de pesquisa	Concluído

Fonte: autores

- Investigar Sistemas Multiagentes, em termos de literatura especializada.
  Status: Material teórico, a respeito de Multiagentes, selecionado e estudado, conferindo aos autores embasamento e fundamentação para a realização dos demais objetivos da pesquisa. O resultado obtido pode ser visto em Referencial Teórico;

- Investigar Sistemas *Matchmaking*, em termos de literatura especializada.

Status: Material teórico, a respeito de *Matchmaking*, além de visitas a fóruns e entendimento de jogos digitais que utilizam Sistemas *Matchmaking* selecionados e estudados, trazendo uma base de conhecimento que auxiliou no decorrer da pesquisa. O resultado obtido pode ser visto em Referencial Teórico;

- Especificar, Projetar, Desenvolver e Documentar uma API, orientada a SMA, capaz de auxiliar na criação de algoritmos de *Matchmaking*.
  Status: Foi realizado o desenvolvimento da aplicação *Multiagent Matchmaking API*, que é uma API de Sistemas *Matchmaking* contruída através do paradigma de Multiagentes, que foi especificada, projetada, desenvolvida e documentada no decorrer desse Trabalho. Os resultados obtidos podem ser vistos em *Multiagent Matchmaking API*, e

- Analisar os Resultados Obtidos, procurando documentar as impressões percebidas, com foco na questão de pesquisa.
  Status: Os resultados coletados através do uso e desenvolvimento da *Multiagent Matchmaking API* foram analisados, através de ciclos de Pesquisa-Ação, e documentados. Os resultados obtidos podem ser vistos em Análise de Resultados.

7.3 Questão de Pesquisa

No decorrer deste trabalho, foram descritos diversos pontos relacionados à construção e à pesquisa de um sistema de *Matchmaking* desenvolvido a partir do paradigma de Multiagentes. Como intenção, o trabalho visa responder a seguinte questão de pesquisa: **Sistemas Multiagentes são aderentes para o desenvolvimento de Sistemas *Matchmaking* de Jogos *Online*?**

Em resposta, e com base no estudo exploratório realizado nesse trabalho, tem-se que o uso de Sistemas Multiagentes é aderente ao desenvolvimento de Sistemas *Matchmaking*. A *Multiagent Matchmaking API* mostrou-se um sistema que faz bom uso de cognição e comunicação, ambas características inerentes ao paradigma de Multiagentes. Além disso, o sistema trouxe pontos positivos, como a fácil escalabilidade do sistema, através do aumento de agentes, que pode ser realizada sob demanda, e também o grande potencial de tolerância a falhas, sendo necessário e pertinente ao se trabalhar com um sistema com comportamentos assíncronos.

Junto à questão de pesquisa, esse trabalho também buscou validar a hipótese de que a solução é adaptável a qualquer jogo. A *Multiagent Matchmaking API* apresenta um sistema de configuração, que possui três critérios, sendo: a quantidade de times, a quantidade de jogadores em um time e a lista de critérios que serão analisados em determinado jogo. Apesar da solução não ser adaptável a qualquer jogo, esses três critérios abrangem a

maioria dos jogos. Dessa forma, pode-se afirmar que a hipótese se apresenta parcialmente satisfeita, considerando que a solução é ADAPTÁVEL para a MAIORIA dos jogos.

7.4 Trabalhos Futuros

Seguem ideias de trabalhos futuros, as quais complementam essa pesquisa de cunho exploratório:

- Evoluir a *Multiagent Matchmaking API* para levar a regionalidade dos jogadores em consideração. Dessa forma, tendo em vista a latência dos jogadores ao formar os *lobbies*;
- Utilizar algoritmos de *machine learning* nos agentes para melhorar o sistema de escolha de jogadores, e
- Evoluir o *HealthAgent* para realizar o controle de agentes sob demanda, derrubando ou instanciando agentes de acordo com a necessidade da fila.

Referências

ABRAGAMES. *A DELEGAÇÃO DO BRAZIL GAMES ESTÁ DE VOLTA PESSOALMENTE PARA A GDC 2022 COM UMA TONELADA DE ESTÚDIOS INCRÍVEIS*. 2021. Disponível em: <https://bit.ly/3EkjyaZ>. Citado na página 23.

AGARWAL, S.; LORCH, J. R. Matchmaking for online games and other latency-sensitive p2p systems. Association for Computing Machinery, New York, NY, USA, v. 39, n. 4, p. 315–326, aug 2009. ISSN 0146-4833. Citado na página 28.

AHMAD, M. O.; MARKKULA, J.; OIVO, M. Kanban in software development: A systematic literature review. p. 9–16, 2013. Citado na página 45.

ALI, I. M. F. *Fortnite Players Stats*. 2021. Disponível em: <https://www.kaggle.com/datasets/iyadali/fortnite-players-stats/metadata>. Citado 3 vezes nas páginas 69, 71 e 73.

ALMAN, J.; MCKAY, D. Theoretical foundations of team matchmaking. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, v. 2, p. 1073–1081, 2017. ISSN 15582914. Citado na página 18.

BELLIFEMINE, F.; CAIRE, G.; GREENWOOD, D. *Developing Multi-Agent Systems with Jade*. [S.l.: s.n.], 2007. 286 p. ISBN 9780470057476. Citado 2 vezes nas páginas 19 e 26.

BLOW, J. Game Development: Harder Than You Think. *Queue*, v. 1, n. 10, p. 28–37, 2004. Citado na página 24.

BRATMAN, M. E. *Intention, Plans, and Practical Reason*. [S.l.]: University of Chicago, 1999. Citado na página 26.

BROOKS, R. A. *Intelligence without representation**. 1991. 139-159 p. Citado na página 26.

CHANDLER, H. M. *The Game Production Toolbox*. [S.l.]: CRC Press, 2020. Citado na página 23.

CLAYPOOL M., D. J. H. G. . O. L. Surrender at 20? matchmaking in league of legends. 2015 ieee games entertainment media conference (gem). 2015. Citado 3 vezes nas páginas 16, 17 e 81.

CSIKSZENTMIHALYI, M. *Flow: The Psychology of Optimal Experience*. [S.l.: s.n.], 1990. Citado na página 17.

DACANAY, R. *How Much Money Did League of Legends Make in 2021?* 2021. Disponível em: <https://www.dbltap.com/posts/how-much-money-did-league-of-legends-make-in-2021-01fr6hfexgdt>. Citado na página 17.

DENG, Q. et al. Globally optimized matchmaking in online games. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*. New York, NY, USA: Association for Computing Machinery, 2021. (KDD '21), p. 2753–2763. ISBN 9781450383325. Citado 2 vezes nas páginas 27 e 28.

DICTIONARY, C. 2022. Disponível em: <https://dictionary.cambridge.org/pt/dicionario/ingles/matchmaking>. Citado na página 26.

DOCKER. 2022. Disponível em: <https://docs.docker.com/get-started/overview/>. Citado na página 36.

DOTA2. 2022. Disponível em: <https://www.dota2.com/home?l=portuguese>. Citado 2 vezes nas páginas 18 e 27.

ELO, A. E. *THE RATING OF CHESSPLAYERS, PAST AND PRESENT*. Second. [S.l.]: ARCO PUBLISHING. INC., 1978. Citado 6 vezes nas páginas 18, 19, 26, 28, 29 e 30.

FATIMA, A.; RASOOL, T.; QAMAR, U. GDGSE: Game Development with Global Software Engineering. *2018 IEEE Games, Entertainment, Media Conference, GEM 2018*, IEEE, p. 288–292, 2018. Citado na página 24.

FIPA. *Agent Communication Language Specifications*. 2002. Disponível em: <http://www.fipa.org/repository/aclspecs.html>. Citado 2 vezes nas páginas 34 e 35.

FIPA. *FIPA Contract Net Interaction Protocol Specification*. 2002. Disponível em: <http://www.fipa.org/specs/fipa00029/SC00029H.html>. Citado na página 55.

FORTNITE. 2022. Disponível em: <https://www.epicgames.com/fortnite/en-US/home>. Citado na página 69.

FOWLER, M. Who needs an architect? *IEEE Software*, v. 20, n. 5, p. 11–13, 2003. Citado na página 52.

FOWLER, M. *Software Architecture Guide*. 2019. Disponível em: <https://martinfowler.com/architecture/>. Citado na página 52.

FREITAS, C. de. *Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico - 2ª Edição*. [S.l.]: Editora Feevale, 2013. Citado 2 vezes nas páginas 38 e 39.

GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. [S.l.]: Pearson Education, 1994. (Addison-Wesley Professional Computing Series). Citado na página 52.

GERHARDT, T.; SILVEIRA, D. *Métodos de Pesquisa*. [S.l.]: PLAGEDER, 2009. (Série Educação a Distância - UFRGS). Citado 2 vezes nas páginas 38 e 39.

GIL, A. *Como elaborar projetos de pesquisa*. [S.l.]: Atlas, 2002. Citado 3 vezes nas páginas 39, 44 e 46.

GIT. 2022. Disponível em: <https://git-scm.com/>. Citado na página 36.

GITHUB. 2022. Disponível em: <https://github.com/about>. Citado na página 36.

GLICKMAN, M. E. The glicko system. *Boston University*, n. June, p. 1–5, 1995. Citado 3 vezes nas páginas 19, 28 e 30.

GRISS, M. L.; HEINEMAN, G. T.; COUNCILL, W. *Software Agents as Next Generation Software Components*. 2001. Disponível em: <www.bottechnology.com>. Citado na página 25.

HALO. *Fórum do Halo Infinite*. 2021. Disponível em: <https://forums.halowaypoint.com/t/matchmaking-unbalance/479361>. Citado na página 49.

HALO. 2022. Disponível em: <https://www.halowaypoint.com/pt-br>. Citado na página 29.

HERBRICH T. MINKA, . T. G. R. Trueskill^{TM}: A bayesian skill rating system, in advances in neural information processing systems. Cambridge, MA, USA: MIT Press, p. 569–576, 2006. Citado 5 vezes nas páginas 17, 19, 29, 30 e 31.

HUIZINGA, J. *Homo ludens: o jogo como elemento da cultura*. [S.l.]: Editora Perspectiva S.A, 1971. 243 p. Citado na página 16.

JADE. 2022. Disponível em: <https://jade.tilab.com/>. Citado 3 vezes nas páginas 34, 35 e 53.

JUPYTER. *Jupyter Notebook*. 2022. Disponível em: <https://jupyter.org/>. Citado na página 37.

KAGGLE. *Kaggle: Your Home for Data Science*. 2022. Disponível em: <https://www.kaggle.com/>. Citado 2 vezes nas páginas 33 e 69.

KISHIMOTO, T. M. *Jogo, brinquedo, brincadeira e a educação*. [S.l.]: Cortez, 1997. Citado na página 16.

KLUSCH, M. Information agent technology for the Internet: A survey. *Data and Knowledge Engineering*, v. 36, n. 3, p. 337–372, 2001. ISSN 0169023X. Citado na página 19.

KULTIMA, A. Developers' perspectives on iteration in game development. *ACADEMICMINDTREK 2015 - Proceedings of the 19th International Academic Mindtrek Conference*, p. 26–32, 2015. Citado na página 24.

LATEX. 2022. Disponível em: <https://www.latex-project.org/about/>. Citado na página 33.

LEGENDS, F. do R. A. 2020. Disponível em: <https://www.reddit.com/r/apexlegends/comments/fx7u0v/long_matchmaking_times/>. Citado na página 49.

LEGENDS, L. of. 2022. Disponível em: <https://www.leagueoflegends.com/pt-br/how-to-play/>. Citado 4 vezes nas páginas 17, 18, 27 e 81.

MCARTHUR, S. D. et al. Multi-agent systems for power engineering applications - part i: Concepts, approaches, and technical challenges. *IEEE Transactions on Power Systems*, v. 22, p. 1743–1752, 11 2007. ISSN 08858950. Citado 2 vezes nas páginas 25 e 26.

MENDELEY. 2022. Disponível em: <https://www.mendeley.com/reference-management/reference-manager>. Citado 2 vezes nas páginas 33 e 44.

MICROSOFT. 2022. Disponível em: <https://www.microsoft.com/pt-br/>. Citado na página 29.

MINKA, T.; CLEVEN, R.; ZAYKOV, Y. *TrueSkill 2: An improved Bayesian skill rating system*. [S.l.], 2018. Citado 3 vezes nas páginas 27, 30 e 31.

MONGODB. 2022. Disponível em: <https://www.mongodb.com/docs/manual/>. Citado na página 36.

NODEJS. *About Node.js*. 2022. Disponível em: <https://nodejs.org/en/about/>. Citado na página 36.

NOGUEIRA, A. T. Um Estudo sobre a Formação de Equipes em Jogos Virtuais Um Estudo sobre a Formação de Equipes em Jogos Virtuais. 2015. Citado na página 16.

OJHA, V. *Top Women Chess Players*. 2020. Disponível em: <https://www.kaggle.com/datasets/vikasojha98/top-women-chess-players>. Citado 3 vezes nas páginas 69, 71 e 75.

OREN, M.; PEDERSEN, S.; BUTLER-PURRY, K. L. Teaching Digital Circuit Design with a 3-D Video Game: The Impact of Using In-Game Tools on Students Performance. *IEEE Transactions on Education*, v. 64, n. 1, p. 24–31, 2021. Citado na página 23.

OVERLEAF. 2022. Disponível em: <https://www.overleaf.com/for/authors>. Citado na página 33.

PELLEGRINI, J.; WAINER, J. Processos de decisão de markov: um tutorial. *Revista de Informática Teórica e Aplicada; Vol. 14, No 2 (2007); 133-179*, v. 14, 12 2007. Citado na página 28.

PEREIRA, L. L.; ROQUE, L. *Avaliação da Experiência de Jogo baseada em Métricas de Participação: o caso do videojogo Fátima*. 2012. Disponível em: <http://playfatima.net>. Citado na página 27.

RABBITMQ. 2022. Disponível em: <https://www.rabbitmq.com/>. Citado na página 35.

REDDIT. 2022. Disponível em: <https://www.reddit.com/>. Citado na página 43.

RIOTGAMES. *Fórum da Riot Games*. 2021. Disponível em: <https://forums.comunidades.riotgames.com/t5/LOL-Mecânicas-de-Jogo/É-dificil-n~ao-tiltar-em-jogos-decididos-pela-sorte-destino/td-p/585180/page/2>. Citado 2 vezes nas páginas 17 e 49.

SADALAGE, P. J.; FOWLER, M. NOSQL Essencial Um Guia Conciso Para o Mundo Emergente da Persistência Poliglota. *Novatec Editora Lda*, p. 216, 2013. Citado na página 53.

SALEN, K.; ZIMMERMAN, E. *Regras do jogo: fundamentos do design de jogos (vol.1)*. [S.l.]: Blucher, 2012. (Principais conceitos). Citado 3 vezes nas páginas 16, 22 e 24.

SALEN, K.; ZIMMERMAN, E. *Regras do jogo: fundamentos do design de jogos (vol.3).* [S.l.]: Blucher, 2012. (Principais conceitos). Citado na página 24.

SCHULZ, K. 2020. Disponível em: <https://www.verizon.com/about/news/how-americans-are-spending-their-time-temporary-new-normal>. Citado na página 49.

SCHWABER, K. *Agile Project Management with Scrum.* [S.l.]: Microsoft Press, 2004. (Best practices). Citado 2 vezes nas páginas 45 e 50.

SIGNIFICADOS. 2022. Disponível em: <https://www.significados.com.br/cognicao/>. Citado na página 18.

SOUZA, M. R. D. A. et al. Games for learning: Bridging game-related education methods to software engineering knowledge areas. *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering and Education Track, ICSE-SEET 2017*, p. 170–179, 2017. Citado na página 23.

TULIO, M. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade.* [S.l.]: Moderna, 2020. Citado na página 52.

VALENTINE, R. *Digital games spending reached $127 billion in 2020.* 2021. Disponível em: <https://www.gamesindustry.biz/articles/2021-01-06-digital-games-spending-reached-USD127-billion-in-2020>. Citado na página 23.

VERIZON. 2022. Disponível em: <https://www.verizon.com/>. Citado na página 49.

VÉRON, M.; MARIN, O.; MONNET, S. Matchmaking in multi-player on-line games: Studying user traces to improve the user experience. *Proceedings of the 24th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV 2014*, p. 7–12, 2014. Citado 4 vezes nas páginas 18, 27, 68 e 81.

WALKER, A. *The evolution of: matchmaking.* 2022. Disponível em: <https://www.readersdigest.co.uk/lifestyle/dating-relationships/the-evolution-of-matchmaking>. Citado na página 26.

WAR, G. of. 2022. Disponível em: <https://gearsofwar.com/pt-br/>. Citado na página 29.

WOOLDRIDGE, M. *An Introduction to MultiAgent Systems - 2nd Edition.* [S.l.: s.n.], 2009. v. 41. 462 p. ISSN 0001-253X. ISBN 978-0-470-51946-2. Citado 4 vezes nas páginas 18, 19, 25 e 26.

XU, M.; YU, Y.; WU, C. Rule designs for optimal online game matchmaking. p. 1055–1062, 2021. Citado 2 vezes nas páginas 28 e 81.