UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO INTERDISCIPLINAR EM TECNOLOGIA DA INFORMAÇÃO

POLLYANA DIAS PAIVA

UM MODELO DE DOCUMENTAÇÃO PARA JOGOS DE ESTRATÉGIA COM ESTUDO DE CASO NO JOGO CLUE SUSPEITOS

PAU DOS FERROS
2025

POLLYANA DIAS PAIVA

UM MODELO DE DOCUMENTAÇÃO PARA JOGOS DE ESTRATÉGIA COM ESTUDO DE CASO NO JOGO CLUE SUSPEITOS

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharela em Interdisciplinar em Tecnologia da Informação.

Orientador: Kennedy Reurison Lopes, Prof. Dr.

Coorientador: José Ferdinandy Silva Chagas, Prof. Me.

PAU DOS FERROS

2025

P142m Paiva, Pollyana Dias.
Um modelo de documentação para jogos de estratégia com estudo de caso no jogo Clue suspeitos / Pollyana Dias Paiva. - 2025.
57 f. : il.

Orientador: Kennedy Reurison Lopes.
Coorientador: José Ferdinandy Silva Chagas.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2025.

1. Engenharia de software. 2. Teoria dos jogos. 3. Documentação. 4. APIs. I. Lopes, Kennedy Reurison, orient. II. Chagas, José Ferdinandy Silva, co-orient. III. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

POLLYANA DIAS PAIVA

UM MODELO DE DOCUMENTAÇÃO PARA JOGOS DE ESTRATÉGIA COM ESTUDO DE CASO NO JOGO CLUE SUSPEITOS

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharela em Interdisciplinar em Tecnologia da Informação.

Defendida em: 11 de Dezembro 2025

BANCA EXAMINADORA

Kennedy Reurison Lopes, Prof. Dr.()
Presidente

Bruno Francisco Xavier, Prof. Dr. (UFERSA)
Membro Examinador

João Batista de Souza Neto, Prof. Dr. (UFERSA)
Membro Examinador

Dedico este trabalho à toda minha família, por terem me apoiado durante este árduo caminho.

AGRADECIMENTOS

Não acredito que finalmente está acabando. Esse trabalho é o resultado do esforço, dedicação e principalmente, das pessoas que estiveram ao meu lado durante essa jornada. Agradeço, antes de tudo, aos meus pais e à minha irmã, que sempre me apoiaram, me acolheram nos momentos difíceis e acreditaram em mim mesmo quando eu duvidava de mim mesma. Sem o carinho e a paciência deles, nada disso seria possível.

Aos meus amigos sou muito grata. Principalmente para a minha melhor amiga que eu poderia ter, minha "Manon" na vida real, sou muito feliz por você ser o meu porto seguro, meu refúgio quando eu mais precisava e por sempre estar ao meu lado em cada momento meu. Para aquela que eu considero como uma segunda mãe, minha querida "Mãe Bruh", agradeço pelo cuidado constante e por sempre me apoiar e por sempre puxar minha orelha quando era necessário. E finalmente ao meu "Fenrys", por estar sempre ao me lado, oferecendo apoio, companhia e seu ombro amigo quando mais necessitei.

Agradeço também aos meus amigos que fiz ao longo do curso, Pedro Vinícius e Pedro Hércules que são os dois "Pedros" do grupo, também agradeço a João Felype, Neilla, Andrey e Davi que foram uma das primeiras amizades que fiz, à o outro Davi que me emprestou a sala do projeto dele para poder escrever e ler sem problemas, à Michely por aturar minhas loucuras, à Ytalo por seu meu amigo mais antigo com uma amizade de 16 anos, também agradecer a Mateus, Vitor e Ângela por serem tão bons amigos e que me aturaram apesar de eu ser uma pessoa chata, também agradecer à todos os outros que eu não citei os nomes, levarei cada um de vocês comigo. E por fim, deixo meu carinho e gratidão aos familiares que me apoiaram de todas as formas, meu tio, minha tia, minha madrinha e agradeço também à Vagna, que sempre me ajudou e ajudou minha família quando a gente mais precisou. Obrigada por acreditarem em mim e por comemorarem cada pequena conquista ao meu lado.

Agradeço à Kennedy e Ferdinandy por terem aceitado serem meus orientador e coorientador respectivamente, à Glaydson por todo apoio fornecido, aos membros da banca, Bruno Xavier e João Batista por terem acreditado no meu trabalho, muito obrigado a todos.

EPÍGRAFE

"Bibliotecas estão cheias de ideias, talvez seja a mais poderosa e perigosa de todas as armas"

(Sarah J. Maas)

RESUMO

O trabalho desenvolveu uma metodologia integrada que combina princípios de Engenhara de Software, documentação de APIs e Teoria dos Jogos para organizar formalmente sistemas interativos baseados em estratégias, utilizando o jogo *Clue Suspeitos* como exemplo prático. Foi mostrado que a lógica estratégica do jogo pode ser convertida em estruturas concretas, como diagramas de sequência, diagramas de classe e a arquitetura C4, garantindo uma estrutura clara, modularidade e coerência entre estados, ações e agentes. A modelagem obtida evidenciou a aplicação de conceitos fundamentais, como alta coesão, baixo acoplamento e separação de responsabilidades, permitindo que a API represente de forma precisa interações que envolvam informações incompletas, decisões condicionais e fluxos estratégicos. Os resultados apontam que a metodologia é eficaz para transformar conceitos abstratos em estruturas formais adequadas ao desenvolvimento de software e suporte a agentes inteligentes capazes de interagir com o sistema e o desenvolvimento de interfaces para testes práticos. Dessa forma, a pesquisa contribui tanto para o avanço de APIs voltadas para jogos estratégicos quanto para pesquisas nas áreas de sistemas interativos, inteligência artificial e modelagem baseada em decisões.

Palavras-chave: engenharia de software; teoria dos jogos; documentação; APIs.

ABSTRACT

This work developed an integrated methodology that combines principles of Software Engineering, API documentation, and Game Theory to formally organize interactive strategy-based systems, using the Clue Suspects game as a practical example. It was shown that the strategic logic of the game can be converted into concrete structures, such as sequence diagrams, class diagrams, and the C4 architecture, ensuring a clear structure, modularity, and coherence between states, actions, and agents. The resulting modeling evidenced the application of fundamental concepts, such as high cohesion, low coupling, and separation of responsibilities, allowing the API to accurately represent interactions involving incomplete information, conditional decisions, and strategic flows. The results indicate that the methodology is effective in transforming abstract concepts into formal structures suitable for software development and support for intelligent agents capable of interacting with the system and the development of interfaces for practical testing. In this way, the research contributes both to the advancement of APIs geared towards strategic games and to research in the areas of interactive systems, artificial intelligence, and decision-based modeling.

Keywords: software engineering; game theory; documentation; APIs.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

APIs *Application Programming Interfaces*

IA *Inteligência Artificial*

UML *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

O avanço da tecnologia e o aumento da capacidade computacional têm impulsionado a criação de jogos cada vez mais complexos e dinâmicos. No contexto dos jogos de estratégia, essa complexidade se manifesta por meio de diversos agentes, cenários distintos e um extenso leque de decisões possíveis. Jogos como "***Civilization***", "***StarCraft***" e "***Total War***" são exemplos claros, onde os jogadores precisam administrar recursos e unidades, além de tomar decisões táticas e estratégicas em tempo real ou por turnos. Esse nível de complexidade exige uma base de software bem estruturada e sólida para o crescimento e desenvolvimento dessas plataformas. Nesse cenário, a elaboração de *Application Programming Interfaces* (APIs) específicas para jogos de estratégia surge como uma alternativa que padroniza funcionalidades, estimula a reutilização de código e facilita a compatibilidade com sistemas de inteligência artificial, contribuindo para um processo de desenvolvimento mais ágil.

A modelagem da jogabilidade (*gameplay*), para jogos de computadores é uma tarefa essencialmente complexa, especialmente quando se leva em conta as decisões e interações que alteram o fluxo do jogo. A teoria dos jogos apresenta um mecanismo formal para analisar o que acontece durante a execução de um jogo, sendo utilizada no design da (*gameplay*) para modelar caminhos e níveis de desafios. Por exemplo, essa teoria oferece estruturas conceituais como árvores de decisão adaptadas e matrizes de (*payoff*), que representam os resultados associados às escolhas dos jogadores em diferentes cenários, possibilitando a modelagem do processo de decisão e os níveis de habilidade ou desafio, como discutido por Taylor *et al.* (2019).

No entanto, uma dificuldade comum em projetos de desenvolvimento de jogos é a ausência de APIs públicas que sejam bem documentadas, uma vez que a documentação de APIs frequentemente apresenta falhas, como a falta de clareza, informações incompletas e desatualizadas como abordado por Meng, Steinhardt e Schubert (2018) e Cummaudo *et al.* (2022). O que acaba prejudicando a produtividade dos desenvolvedores, um desafio que se intensifica em sistemas complexos, como os jogos de estratégica.

Segundo Sommerville (2019), a Engenharia de Software é um campo que abrange todos os aspectos relacionados à criação de software, destacam-se a modularidade, a documentação e a clareza na definição de responsabilidades. Esses elementos cruciais para assegurar a manutenibilidade, a escalabilidade e a compreensão de sistemas complexos. A aplicação desses princípios na criação de APIs garante não apenas a funcionalidade técnica, mas também a facilidade de uso para outros desenvolvedores e pesquisadores.

Nesse sentido, a arquitetura modular de software, que divide o software em módulos independentes, é fundamental para a manutenção, o reúso de código e a escalabilidade do sistema, como discutido por Nery e Silva (2022). A adoção de arquiteturas com baixo acoplamento, como os microsserviços, também é crucial para a escalabilidade e a manutenibilidade em sistemas complexos, conforme apontado por Almeida (2024), esse tipo de arquitetura pode trazer diferentes graus de complexidade, principalmente em situações onde diversos serviços são implementados de forma independente. A integração entre tais serviços, assim como a troca de informações e a colaboração entre eles, se transformam em aspectos extras de complexidade que precisam ser levados em conta com atenção no projeto arquitetural.

Além disso, Tonini, Carvalho e Spínola (2008) ressaltam a importância de processos estruturados e maduros no desenvolvimento de software, observando que a maturidade organizacional é fundamental para garantir a sustentabilidade e a evolução contínua de sistemas computacionais. A elaboração de uma API baseada em boas práticas de engenharia contribui diretamente esses objetivos, possibilitando a reutilização da estrutura em diferentes jogos e projetos.

A ausência ou a inadequação da documentação é um dos principais obstáculos à manutenção e à compreensão de sistemas, dificultando a adaptação por terceiros e impactando a produtividade dos desenvolvedores, conforme amplamente investigado por Robillard e Deline (2011) em estudos de campo sobre obstáculos de aprendizado de API. Esse aspecto é particularmente importante na construção de APIs voltadas para jogos de estratégias, nos quais a complexidade das interações e das regras exige um elevado nível de clareza e precisão na descrição dos componentes. Para APIs, a documentação é a "porta de entrada", definindo expectativas claras, reduzindo o tempo de *onboarding* e capacitando outros a inovar, sendo considerada um dos setes princípios-chaves do design de API, conforme destacado por Araujo e Chagas (2023).

A proposta deste trabalho é desenvolver uma metodologia voltada à criação de APIs para jogos de estratégia, com base nos fundamentos da Teoria dos Jogos. Essa teoria, comumente aplicada na modelagem de conflitos e negociações, oferece uma estrutura conceitual robusta para descrever cenários (estados de jogo), ações estratégicas (decisões dos jogadores) e agentes inteligentes (participantes do jogo). Assim, a aplicação dos princípios da Teoria dos Jogos facilita a representação precisa e flexível das interações estratégicas entre os jogadores, contribuindo na criação de APIs mais robustas e adaptáveis.

Dessa forma, este trabalho combina os princípios da Engenharia de Software com a

Teoria dos Jogos para propor uma metodologia abrangente para a construção de APIs orientadas a jogos de estratégia. Para validar a metodologia proposta, será documentada uma API para o jogo *Clue Suspeitos*, que servirá como um estudo de caso central para demonstrar a aplicação de seus princípios.

1.1 Motivação

Este trabalho está inserido no âmbito do projeto de pesquisa "*Aplicações da Teoria dos Jogos na Resolução de Problemas Complexos: Estratégias Ótimas em Cenários Competitivos e Cooperativos*", que procura investigar e aplicar os princípios da teoria em sistemas computacionais interativos. A ideia de criar uma metodologia para o desenvolvimento de APIs voltadas para jogos de estratégia surge da necessidade de padronizar e facilitar a criação desses jogos, especialmente em contextos que envolvem decisões estratégicas entre vários agentes.

Jogos de estratégia envolvem decisões em situações incertas, com informações incompletas e interações entre jogadores que tentam alcançar metas individuais com base em raciocínio lógico e dedução. Essas características tornam tais jogos ótimos para a análise do comportamento estratégico e para a criação de agentes inteligentes. Contudo, uma dificuldade recorrente em projetos voltados ao desenvolvimento desses jogos é a escassez de APIs públicas, bem documentadas, que apresentem de forma clara e modular a lógica que rege as interações do jogo.

Apesar da popularidade e da complexidade se tornando cada vez mais crescente dos jogos de estratégia, há uma falta evidente de APIs especializadas que possam atender às suas necessidades. Uma APIs ideal deve ser capaz de lidar com as complexidades que são características dos jogos de estratégias, como múltiplos agentes, cenários dinâmicos e decisões estratégicas complexas. Isso cria uma lacuna significativa para desenvolvedores e pesquisadores que desejam criar jogos de estratégia atraentes e sólidas. Portanto, o principal objetivo deste trabalho é desenvolver uma estrutura dedicada que ajude na criação de APIs que sejam adaptadas às características dos jogos de estratégia.

Trabalhos como os de Taylor *et al.* (2019) já apontam a complexidade de modelar e documentar a jogabilidade em jogos de computador, o que se estende à criação de APIs robustas. Além disso, a literatura geral sobre documentação de APIs, como demonstrado por Meng, Steinhardt e Schubert (2018) e Cummaudo *et al.* (2022), frequentemente revela deficiências como incompletude, falta de clareza e desatualização, impactando a produtividade dos desenvolvedores e a adoção dessas interfaces, um cenário que se agrava em sistemas complexos e com agentes

inteligentes.

Ao apresentar uma metodologia voltada para o desenvolvimento de APIs fundamentada em princípios da Teoria dos Jogos, este trabalho busca oferecer uma estrutura genérica e adaptável, capaz de descrever cenários (estados de jogos), ações possíveis (tomadas de decisão) e agentes (jogadores ou sistemas inteligentes) para diversos jogos de estratégia, promovendo a reutilização, a integração e a expansão desses sistemas. Essa abordagem será demonstrada através da documentação de uma API específica para um jogo de estratégia, servindo como exemplo concreto para que outros desenvolvedores apliquem os mesmos conceitos em jogos semelhantes.

A relevância de uma abordagem organizada e documentada é amplamente discutida na literatura da Engenharia de Software. Sommerville (2019) menciona que práticas como modularização e documentação são fundamentais para garantir que os sistemas permaneçam compreensíveis, reutilizáveis e adaptáveis ao longo do tempo. A modularidade de software, que permite a decomposição de sistemas em partes independentes e reutilizáveis, é crucial para a organização lógica e a documentação da arquitetura de um sistema, inclusive com o uso de diagramas UML, como destacado por Bernhardt *et al.* (2023). Robillard e Deline (2011) também enfatizam que a ausência de documentação clara é um dos maiores obstáculos à manutenção de sistemas, o que impacta diretamente sua sustentabilidade.

Adicionalmente, processos bem estruturados e amadurecidos no desenvolvimento de software é crucial para a qualidade e sustentabilidade de produtos, conforme observado por Tonini, Carvalho e Spínola (2008). No contexto de jogos de estratégia, estabelecer um processo claro para a elaboração de APIs é fundamental para assegurar que novas versões ou adaptações possam ser implementadas sem comprometer a integridade do sistema.

Do ponto de vista técnico, a documentação da API para o jogo Clue será elaborada com atenção especial à clareza, divisão de responsabilidades e conformidade com padrões reconhecidos de desenvolvimento e documentação amplamente aceitas. A proposta inclui modelagem da lógica do jogo utilizando diagramas UML, como casos de uso, sequência e classes, juntamente com descrições funcionais das ações, estados e regras. Embora a implementação da API não faça parte deste projeto, a documentação será detalhada e completa, incorporando todos os elementos essenciais para qualquer desenvolvedor possa viabilizar sua futura implementação de forma autônoma.

Em suma, a motivação deste trabalho é estabelecer uma base metodológica que ajude na criação de sistemas interativos voltados para os jogos de estratégia, promovendo a conexão entre

a Teoria dos Jogos e a Engenharia de Software. A documentação da API de jogo de estratégia serve como um exemplo representativo, permitindo a validação da metodologia proposta e atuando como referência para desenvolvimentos futuros no âmbito do projeto de pesquisa.

1.2 Objetivos Gerais

Desenvolver uma metodologia estruturada para o desenvolvimento de APIs destinadas a jogos de estratégia, fundamentada na Teoria dos Jogos e nas melhores práticas de Engenharia de Software. Essa metodologia será validada através do estudo de caso da modelagem e e especificação detalhada da mecânica do *Clue Suspeitos*.

1.3 Objetivos Específicos

1. Investigar os princípios da Teoria dos Jogos aplicáveis ao desenvolvimento de sistemas interativos com múltiplos agentes, com foco em decisões estratégicas e interações competitivas e cooperativas.
2. Identificar os requisitos funcionais e modelar a mecânica do jogo *Clue Suspeitos*, considerando cenários, escolhas e perfis de agentes.
3. Projetar a arquitetura modular da API, estabelecendo responsabilidades, fluxos de informação, transições de estado e pontos de extensão.
4. Elaborar uma documentação da modelagem da API, utilizando padrões técnicos e visuais reconhecidos (como diagramas UML, descrições de modelo e estruturação temática).
5. Disponibilizar a documentação da modelagem da API como um recurso de referência aberto para desenvolvedores e pesquisadores interessados no desenvolvimento de jogos com agentes inteligentes.

2 FUNDAMENTAÇÃO TEÓRICA

O desenvolvimento de APIs para jogos de estratégia traz à tona desafios específicos que estão ligados à organização do jogo, à interação entre diversos agentes e à necessidade de uma documentação clara que possibilite uma adequada integração de agentes inteligentes. A ausência de normas bem definidas para organizar e documentar esse tipo de API torna o trabalho dos desenvolvedores mais difícil, favorece interpretações confusas e prejudica a aplicação de comportamentos estratégicos consistentes. Diante dessa situação, é essencial estabelecer uma base teórica sólida que guie a modelagem e a documentação.

A fundamentação teórica deste trabalho é fundamentada em quatro pilares principais: Engenharia de Software, Documentação de Código e APIs, Modularização e Teoria dos Jogos. Esses pilares fornecem os conceitos e métodos necessários para desenvolver uma abordagem efetiva direcionada à documentação de APIs em jogos de estratégia, *Clue Suspeitos*. Cada um desses campos será examinado detalhadamente, oferecendo definições, aplicações práticas e contribuições relevantes para a elaboração de sistemas interativos e coerentes que envolvem múltiplos agentes.

2.1 Teoria dos Jogos

2.1.1 Fundamentos Conceituais

A Teoria dos Jogos é um ramo da matemática aplicada que estuda comportamentos estratégicos entre agentes racionais, onde as escolhas de cada participante impactam diretamente nos resultados dos outros. Conforme menciona Francez (2017), essa teoria surgiu com um foco na economia, mas rapidamente se expandiu para vários setores, dada sua relevância em ambientes que demandam decisões estratégicas, como política, negociações, biologia evolutiva e ciência da computação.

Neumann e Morgenstern (1944) foram os pioneiros no desenvolvimento formal da teoria com a clássica obra ***Theory of Games and Economic Behavior*** (1944). Eles conceituaram os jogos como representações matemáticas de situações de conflito e colaboração, com a capacidade de indicar as melhores soluções ou descartar estratégias ineficazes. Como observado por Vitoriano, Neto e Spers (2009), a Teoria dos Jogos fornece uma compreensão abrangente das interações estratégicas reais e descarta jogadas que não produzem resultados favoráveis.

2.1.2 Estratégias e Equilíbrios

Um princípio fundamental da Teoria dos Jogos é o conceito de estratégia, que é o conjunto de escolhas que um jogador pode efetuar para alcançar o resultado mais favorável. Conforme Sartini *et al.* (2004), uma estratégia dominante é aquela que garante o melhor resultado, não importando a escolha do oponente, enquanto o equilíbrio de Nash é uma situação na qual nenhum jogador tem vantagem em alterar sua escolha de forma unilateral, desde que os demais mantenham suas decisões.

Neste trabalho, o equilíbrio de Nash é apresentado como uma referência teórica, e não como um sistema computacional que foi realmente implementado na API. A abordagem não envolve o cálculo explícito de equilíbrios, mas utiliza os princípios da Teoria dos Jogos para orientar a modelagem de decisões e interações entre agentes, o que é adequado a jogos de informação incompleta e não determinísticas, semelhantes ao *Clue Suspeitos*.

Esse conceito é fundamental para a análise de jogos de estratégia, pois representa um ponto de estabilidade em que nenhum jogador pode melhorar seu resultado individualmente sem prejudicar a cooperação ou o equilíbrio do sistema como um todo. Em contextos computacionais, o equilíbrio de Nash serve como base para o desenvolvimento de agentes inteligentes capazes de prever e reagir às ações dos demais participantes, o que é essencial para a criação de sistemas interativos robustos e realistas.

Um exemplo clássico de equilíbrio de Nash é o "Dilema do Prisioneiro", no qual dois jogadores devem escolher entre cooperar ou trair sem saber a decisão do outro. O equilíbrio ocorre quando ambos optam por trair, pois nenhum deles pode melhorar seu resultado individual mudando sua decisão unilateralmente, mesmo que essa escolha não seja a ideal para o grupo como um todo. Esse cenário demonstra como o equilíbrio de Nash pode representar soluções estáveis, ainda que sub-ótimas, em situações estratégicas.

Ademais, a diferença entre estratégias puras e mistas (como propostas por Waldegrave em 1713 e estudadas posteriormente por Nash e Zermelo) evidencia como diversas opções de decisão podem ser rigorosamente modeladas, algo que se torna vital em ambientes interativos e digitais.

É importante mencionar que o *Clue Suspeitos* é classificado como um jogo de não soma zero, onde as decisões de um jogador não resultam necessariamente em perdas diretas para os outros, e possui informações incompletas, pois os participantes têm conhecimentos diferentes durante o jogo. Essas características tornam pertinente a adoção da Teoria dos Jogos como

base conceitual para organizar as decisões e os agentes, sem a necessidade de resolver o jogo matematicamente.

2.1.3 Jogos de Dedução e Informação Incompleta: o Caso de *Clue Suspeitos*

Os jogso de estratégia que se baseiam em dedução são exemplos clássicos de ambientes com informação incompleta, onde os jogadores tomam decisões táticas a partir de informações parciais sobre a situação do jogo e o comportamento dos outros jogadores. Esse tipo de cenário é frequentemente abordado na Teoria do Jogos, especialmente em contextos em que os agentes precisam descobrir informações ocultas baseadas em sinais indiretos e ações que podem ser observadas, conforme apontado por Figueiredo (1994) e Francez (2017).

Nesse contexto, o jogo *Clue Suspeitos*®, que se inspira no clássico *Clue* e em jogos do tipo Detetive, possui uma estrutura que exemplifica claramente essas características. O objetivo principal é identificar de forma precisa os elementos que formam um crime fictício: a arma utilizada, o local ocorrido e o suspeito envolvido. Esses elementos estão representados por um conjunto de 15 cartas, divididas em três categorias: quatro tipos de armas, cinco locais e seis suspeitos. No início de cada partida, uma carta de cada categoria é escolhida aleatoriamente e removida, criando assim o conjunto secreto que representa o crime. As 12 cartas restantes são misturadas e distribuídas igualmente entre os participantes, que podem variar entre dois a quatro jogadores HASBRO (2018).

A dinâmica do jogo é baseada em quatro mecânicas principais: realizar perguntas, responder perguntas, anotar e acusar.

- **Realizar Pergunta:** permite que um jogador questione o jogador seguinte sobre a posse de combinações específicas de cartas, abrangendo diferentes categorias. Se o jogador interrogado não tiver nenhuma das cartas mencionadas, a pergunta é passada aos outros jogadores em ordem, até que todos tenham sido questionados. Essa interação caracteriza um processo gradual de coleta de dados, que é típico em jogos de dedução estratégica Taylor *et al.* (2019).
- **Responder Pergunta:** exige que o jogador ao qual foi feita a pergunta forneça uma reposta sincera, de acordo com suas cartas de evidência. Se ele tiver apenas uma das cartas abordadas, deve exibi-la apenas para o jogador que fez a pergunta. Se o jogador tiver mais de uma das cartas mencionadas, ele pode escolher qual delas revelar, sem indicar a existência das demais. Se não tiver nenhuma carta relevante, deve informar que não pode

ajudar. Embora as informações que foram reveladas sejam privadas, a ação de perguntar, a identidade dos jogadores e o resultado da resposta são informações públicas, acessíveis a todos os jogadores. Esse mecanismo aumenta a assimetria de informações entre os jogadores, um aspecto frequentemente analisado em modelos estratégicos pela Teoria dos Jogos Francez (2017).

- **Anotar:** funciona como um auxílio mental para o processo de raciocínio. Cada jogador receber um conjunto extra de cartas ou marcadores que simbolizam todos os elementos disponíveis no jogo, facilitando o registro e a organização das informações obtidas ao longo da partida. Essa abordagem está vinculada à necessidade de ordenar dados incompletos para auxiliar na tomada de decisões racionais, conforme mencionado por Figueiredo (1994) em contextos de estratégia.
- **Acusar:** marca o ponto culminante do jogo. Na sua vez, o jogador pode escolher fazer uma acusação em vez de fazer uma pergunta. Nesse caso, todos os jogadores apresentam um palpite sobre o crime, criando grupos de três cartas que são comparadas ao conjunto secreto (confidencial). O ganhador é aquele que conseguir adivinhar a combinação correta, seguindo a ordem de quem iniciou a acusação. Esse momento resume o processo de decisão sob incertezas, onde os jogadores pesam riscos e vantagens com base nas informações reunidas durante o jogo, conforme discutido por Taylor *et al.* (2019).

A partir desses conjunto de mecânicas, podemos notar que o *Clue Suspeitos* cria um cenário estratégico no qual as escolhas dos jogadores são influenciadas por informações incompletas, observações indiretas e deduções sobre as ações dos outros participantes. Essa configuração está em harmonia com os conceitos da Teoria dos Jogos, especialmente no que se refere à identificação de agentes, estados de jogo e ações possíveis. É importante destacar, entretanto, que neste estudo tais conceitos são utilizados como uma base conceitual, sem a aplicação de modelos matemáticos rigorosos ou soluções de equilíbrio, servindo como fundação para a organização e documentação de APIs destinadas a jogos de estratégia.

2.1.4 Aplicações Computacionais

No contexto computacional, a Teoria dos Jogos é fundamental para desenvolver sistemas autônomos e inteligentes, simulações de mercado e jogos interativos. Como destacado por Figueiredo (1994), a teoria oferece instrumentos para modelar e solucionar questões de alocação de recursos, formação de coalizões e interações estratégicas entre agentes em ambientes

descentralizados.

A formalização matemática de jogos, como os jogos de soma-zero de barganha, é crucial para a implementação de APIs e sistemas onde agentes precisam fazer escolhas em tempo real com base em informações incompletas e dinâmicas. Um "jogo de soma-zero de barganha"pode ser ilustrado por uma negociação onde o ganho de um participante corresponde diretamente à perda do outro, como em um leilão de preço único, onde o lance vencedor representa um ganho para o vendedor e uma perda (valor pago) para o comprador, e vice-versa. Francez (2017) enfatiza que a Teoria dos Jogos estimula o raciocínio lógico e proporciona um modelo estruturado para examinar comportamentos estratégicos com um elevado nível de abstração matemática.

Integrar a Teoria dos Jogos com os fundamentos da Engenharia de Software aprimora a criação de sistemas estratégicos, incluindo jogos interativos, plataformas de simulação e aplicações de inteligência artificial. A precisão na identificação dos agentes, contextos e recompensas torna a documentação de APIs com objetivos específicos mais clara. Essa integração permite uma representação mais formal de situações complexas e ajuda a estabelecer decisões automatizadas, favorecendo maior previsibilidade, reusabilidade e controle na modelagem de jogos computacionais.

Decisões automatizadas, nesse contexto, referem-se a escolhas tomadas por agentes de software (como inteligências artificiais em jogos) sem intervenção humana direta, baseadas em algoritmos e modelos derivados da Teoria dos Jogos para otimizar seus resultados em cenários estratégicos. Um trabalho que discute a aplicação da Teoria dos Jogos no design de jogos de computador, abordando a modelagem de jogabilidade e níveis de dificuldade é o de Taylor *et al.* (2019).

2.2 Engenharia de Software

2.2.1 Conceitos Básicos

A Engenharia de Software, conforme definida por Sommerville (2019), é "uma disciplina de engenharia que se ocupa de todos os aspectos da produção de software, desde as fases iniciais até a manutenção do sistema após sua entrega". Esse campo abrange métodos e práticas que visam garantir a qualidade, a confiabilidade e a manutenibilidade dos sistemas de software ao longo do tempo.

2.2.2 Princípios de Qualidade e Sustentabilidade

Um dos focos centrais da Engenharia de Software é assegurar que os sistemas sejam fáceis de manter, escaláveis e compreensíveis. Tonini, Carvalho e Spínola (2008) enfatizam que a implementação de processos maduros e organizados possibilita melhorias contínuas, garantindo qualidade e longevidade aos sistemas. Isso é crucial na criação de APIs para jogos de estratégia, onde alterações e adaptações frequentes são esperadas. Além disso, esses autores ressaltam a relevância da maturidade organizacional como fundamento para o sucesso duradouro dos sistemas de software.

2.2.3 Aplicação da Engenharia em APIs

No que se refere a APIs, os princípios da Engenharia de Software são essenciais na construção de interfaces sólidas e reutilizáveis. A adequada separação de funções, a definição de requisitos, o uso de padrões como UML e a adoção de boas práticas de versionamento e testes fazem parte de uma abordagem necessária para garantir que uma API documentada seja acessível e útil para outros desenvolvedores. Essa estratégia, além de elevar a qualidade técnica da API, facilita sua aceitação em ambientes educacionais, comerciais e de pesquisa.

2.3 Arquitetura de Software

2.3.1 Definições e Importância

A Arquitetura de Software é a estrutura fundamental de um sistema de software, abrangendo seus componentes, seus relacionamentos, os princípios que guiam seu design e sua evolução ao longo do tempo. Ela define como o sistema é organizado e como suas partes interagem para cumprir os requisitos funcionais e não funcionais. Uma arquitetura bem definida é crucial para a qualidade, manutenibilidade, escalabilidade e desempenho de sistemas complexos Nery e Silva (2022).

2.3.2 Modularidade: Coesão e Acoplamento

A modularidade é um princípios central da arquitetura de software, que consiste em segmentar um sistema em partes menores e independentes, chamadas módulos. De acordo com Sommerville (2019), a modularização permite maior controle sobre a arquitetura do sistema,

simplifica sua evolução e permite que diferentes seções sejam trabalhadas de maneira isolada e eficaz. Um sistema modular frequentemente apresenta maior robustez, pois possibilita isolar falhas, testar módulos de foram individual e realizar modificações sem grandes repercussões no restante da aplicação.

A construção de módulos deve focar em garantir alta coesão (nível de ligação entre componentes internos de um módulo) e baixo acoplamento (dependência entre módulos distintos). Ao minimizar o acoplamento, o risco de falhas em cadeia é reduzido e a flexibilidade da arquitetura é ampliada. A união de alta coesão e baixo acoplamento é vista como um dos principais objetivos de design em sistemas modulares, impactando diretamente a qualidade da arquitetura, conforme discutido por Luque (2010).

2.3.3 Reutilização e Expansibilidade em APIs de Jogos

Ao aplicar essas ideias em jogos de estratégia, como o *Clue Suspeitos*, é viável criar APIs com funções claramente definidas, como iniciar um jogo, realizar uma jogada ou checar o estado da partida. Cada uma dessas funcionalidades pode ser documentada de maneira independente, o que facilita a compreensão e a manutenção, possibilitando que os mesmos módulos sejam utilizados em diversos jogos ou sistemas facilitando a ampliação do projeto sem prejudicar sua estrutura original, algo fundamental para fins educacionais ou pesquisas sobre agentes inteligentes. Ter módulos bem definidos e documentados diminui a repetição de esforços e aumenta a consistência no desenvolvimento, sendo um resultado direto da aplicação dos princípios de arquitetura de software que esta metodologia propõe.

2.4 Documentação de Códigos e APIs

2.4.1 Papel da Documentação

A documentação de um código é uma das atividades mais importantes no ciclo de vida de desenvolvimento de software. Robillard e Deline (2011) enfatizam que a falta de documentação apropriada afeta de forma prejudicial a manutenção e o progresso do sistema. Esse problema se intensifica em projetos colaborativos, onde vários programadores necessitam compreender a lógica por trás do código. Uma boa documentação também contribui em auditorias, análises de código e procedimentos de integração contínua.

2.4.2 Documentação de APIs

Para as APIs, a documentação assume um papel ainda mais fundamental. Araujo e Chagas (2023) ressaltam que, apesar de frequentemente ser deixada de lado, a documentação é essencial para assegurar o êxito e o uso eficiente das interfaces por usuários externos. Deve incluir não apenas descrições funcionais, mas também exemplos práticos, fluxos e estruturas de dados. A clareza e a abrangência dessas informações afetam diretamente a adaptabilidade e a expansão das APIs. Para as APIs, as documentação é a "porta de entrada"estabelecendo expectativas nítidas, diminuindo o período de integração e permitindo que outros façam inovações, sendo vista como um dos sete princípios fundamentais do design de API Araujo e Chagas (2023).

2.4.3 Boas Práticas e Ferramentas

Utilizar ferramentas como *Swagger*, *Redoc*, *Sphinx* ou *JSDoc* pode facilitar a apresentação da API. Adicionalmente, a utilização de *docstrings*, comentários bem organizados, diagramas (como os da UML, que são cruciais para a organização lógica e a documentação da arquitetura de um sistema modular Bernhardt *et al.* (2023)) e manuais de uso são componentes que enriquecem uma documentação eficaz. A adoção de padrões assegura que outros desenvolvedores consigam compreender e modificar a API em novos ambientes. Essas práticas tornam a API disponível para um público técnico variado, incentivando sua aceitação e manutenção.

2.4.4 Impacto na Reusabilidade

Elaborar uma documentação detalhada da API *Clue Suspeitos* possibilitará sua utilização em outros projetos e facilitará o desenvolvimento de agentes autônomos que possam interagir com a lógica do jogo. Isso diminui a necessidade de retrabalho, favorece a uniformidade e estimula a criação de soluções originais que utilizem a mesma estrutura lógica. Nesse cenário, a documentação funciona como um elo entre a lógica interna da aplicação e sua utilização externa, servindo como uma ferramenta para disseminação e colaboração.

3 TRABALHOS RELACIONADOS

Esta seção apresenta uma revisão de pesquisas acadêmicas que exploram a interseção entre Engenharia de Software e Teoria dos Jogos. A análise desses trabalhos revela a variedade e a flexibilidade com a qual os conceitos da Teoria dos Jogos são utilizados para enfrentar desafios em diferentes etapas do ciclo de vido do desenvolvimento de software. Além de apresentar um panorama sobre como a área tem evoluído. O capítulo fornece uma análise organizada das abordagens principais, permitindo uma compreensão do estado atual e contextualizando a relevância deste estudo.

Ao longo do capítulo, são discutidas pesquisas que aplicam a Teoria dos Jogos na gestão de projetos e processos de software, explorando como estratégias cooperativas e competitivas auxiliam no alinhamento do interesse e na tomada de decisões em equipes de desenvolvimento. Também são explorados estudos que utilizam modelos estratégicos na engenharia de sistemas e na cibersegurança, especialmente no que diz respeito a modelagem das interações entre agentes adversários. Em seguida, são apresentados estudos voltados a áreas técnicas específicas na Engenharia de Software, como teste, verificação, definição de requisitos entre outras atividades onde o comportamento estratégico dos participantes impacta diretamente os resultados.

Além disso, o capítulo une pesquisas relacionadas à gamificação e ao desenvolvimento de APIs para ambientes educacionais, destacando a maneira como as mecânicas e a dinâmicas de jogos podem ser estruturadas como interfaces reutilizáveis. Também são analisadas investigações sobre *frameworks* e APIs voltadas a jogos estratégicas, que são essenciais para o progresso de agentes inteligentes e para a padronização de contextos experimentais. Por último, são discutidas contribuições a respeito do ajuste dinâmico da dificuldade em jogos digitais e investigações sobre a documentação de APIs, destacando a relevância de estruturas claras e exemplos apropriados para facilitar o uso correto dessas interfaces.

Ao organizar essas perspectivas, esta revisão situa a proposta da pesquisa, que, embora se relacione com os trabalhos anteriores, se volta a uma área com grande potencial para a exploração: a documentação de APIs para jogos de estratégia, organizada de forma a proporcionar um suporte eficaz ao desenvolvimento e à integração de agentes inteligentes.

3.1 Teoria dos Jogos na Gestão de Projetos e Processos

Nesta seção são apresentadas pesquisas que utilizam a Teoria dos Jogos como suporte para aprimorar processos de desenvolvimento e a dinâmica de interação entre equipes de software. Esses estudos destacam de que forma os conceitos estratégicos podem ajudar a equilibrar interesse, na medição de conflitos e na melhoria da eficiência durante o planejamento e a execução de projetos, contribuindo paras as decisões mais justas e colaborativas no contexto da Engenharia de Software.

No artirgo de Yilmaz e O'Connor (2010) ***"Maximizing the Value of the Software Development Process by Game Theoretic Analysis"***, apresentam a ideia de design de mecanismos econômicos dentro do processo de desenvolvimento de software. Esta pesquisa é especialmente importante pois trata o desenvolvimento como uma atividade econômica dentro de um ecossistema de troca de informações, utilizando a Teoria dos jogos para aprimorar a alocação de recursos e a coordenação das equipes.

A principal contribuição consiste na proposta de mecanismos que alinham os incentivos pessoais com os objetivos gerais do projeto, buscando maximizar o valor gerado. Apesar dessa perspectiva ser importante, ela difere do projeto em pauta, que não foca na administração econômica, mas sim em utilizar a teoria para estruturar a lógica de um jogo estratégico, permitindo a elaboração de documentação e o desenvolvimento de agentes inteligentes.

A pesquisa de Ajienka (2015), ***"An Application of Game Theory in the Resolution of Software User Requirements Conflicts"***, apresenta um exemplo prático da utilização da Teoria dos Jogos para resolver conflitos de requisitos entre usuários e partes interessadas. A autora detalha cuidadosamente como os jogos cooperativos podem ser aplicados para fomentar a colaboração e alcançar consensos entre grupos com necessidades conflitantes.

A principal contribuição é a sugestão de uma ferramenta de negociação auxiliada por computador que integra os princípios de *payoffs* para demonstrar o impacto das decisões. O trabalho de Ajienka se destaca por seu foco na gestão de projetos e na resolução de conflitos, o que o torna distinto do estudo, que utiliza a Teoria dos Jogos para representar a lógica interna de um jogo, com o objetivo de documentar APIs.

No artigo de Bildirici, Codal e Medeni (2024) ***"Using Agile Story Points and Game Theory Together: Better Software Planning and Development in Agile Software Development"***, analisaram a combinação da Teoria dos Jogos com metodologias ágeis. Eles sustentam que a estimativa de *Story Points* pode ser vista como um jogo estratégico, onde a Teoria dos Jogos é

capaz de lidar com vieses e erros, aprimorando a organização e a dinâmica da equipe.

Sua participação é evidente na criação de planos baseados em dinâmicas de jogos como o Leilão de Vickrey e o Jogo da Caça ao Cervo para aprimorar as estimativas e resolver conflitos. O foco, portanto, é na administração e avaliação de projetos ágeis, enquanto a pesquisa utiliza a Teoria dos Jogos para organizar a lógica interna do jogo e sua documentação por meio de API, concentrando-se na interação entre agentes inteligentes.

3.2 Teoria dos Jogos em Engenharia de Sistemas e Segurança Cibernética

Nesta seção são reunidas pesquisas que aplicam a Teoria dos Jogos em cenários com maior complexidade técnica, incluindo sistemas distribuídos, engenharia de requisitos avançados e segurança digital. Os estudos discutidos mostram como agentes racionais (cooperativos ou adversários) podem ser representados matematicamente para prever comportamentos, analisar ameças e apoiar decisões em ambientes altamente dinâmicos.

Em, ***"A Survey of Game Theory as Applied to Network Security"*** Roy *et al.* (2010), realizam uma análise detalhada sobre a utilização da Teoria dos Jogos na proteção de redes. Este estudo é uma fonte importante, pois estabelece uma base para a modelagem matemática de problemas de segurança envolvendo agentes com objetivos muitas vezes opostos, como invasores e defensores.

A principal contribuição reside na sua classificação, que organiza as soluções disponíveis e fornece um esquema para examinar taticamente diferentes alternativas de ameaças e antecipar resultados.Contudo, a atenção de Roy *et al.* (2010) é unicamente voltada para a proteção de redes. A pesquisa em questão, por outro lado, foca em estruturar a lógica interna de jogos estratégicos, lidando com um campo de aplicação e um grau de abstração diferentes.

No estudo de Yazdani *et al.* (2018) intitulado ***"A Game Theory Perspective on Requirement-Based Engineering Design"***, empregaram a Teoria dos Jogos não cooperativos para explorar o design de engenharia orientado a requisitos. Este trabalho é significativo, pois demonstra que a Teoria dos Jogos pode prever os resultados das decisões em ambientes de design, onde diversos tomadores de decisões tentam maximizar seus próprios interesses.

A principal contribuição é fornecer uma base teórica que explica como a busca por otimização individual pode levar a resultados insatisfatórios para o projeto em sua totalidade. Este estudo, por outro lado, se distingue ao utilizar a Teoria dos Jogos para representar a dinâmica e alógica interna do jogo estratégico, visando documentar a API, e não focando no processo de

elaboração de requisitos para uma solução de engenharia.

No artigo de Grigoryan e Collins (2021) ***"Game Theory for Systems Engineering: A Survey"***, forneceram uma análise minuciosa da Teoria dos Jogos em Engenharia de Sistemas. Este trabalho é uma referência significativa ao destacar a importância da teoria em sistemas complexos com múltiplos decisores.

A contribuição oferece uma visão nítida e completa sobre como a Teoria dos Jogos pode auxiliar na compreensão de interações e escolhas em circunstâncias incertas, destacando sua capacidade de modelar sistemas e reconhecer equilíbrios. no estando, o trabalho apresentado permanece em um patamar de uma visão geral. Em contrapartida, o projeto mencionado é o mais focado e especializado, sugerindo uma abordagem específica, para a documentação de APIs destinadas a jogos de estratégias, o que o diferencia de uma análise mais abrangente.

Mais recentemente, Khalid, Al-Kadhimi e Singh (2023), no artigo intitulado ***"Recent Developments in Game-Theory Approaches for Detection and Defense Against Advanced Persistent Threats (APTs)"***, conduziram uma análise detalhada sobre como a Teoria dos jogos é utilizada para investigar situações complexas relacionadas à segurança cibernética. O estudo contribui ao enfatizar as tendências e a aplicação de diferentes tipos de jogos (estáticos, dinâmicos, Markovianos, Bayesianos, etc.) para modelar as interações estratégicas entre os atacantes e os defensores.

A conexão com a pesquisa está na aplicação da Teoria dos Jogos em sistemas digitais complexos que possuem agentes adversários. No entanto, a área de segurança cibernética, com seu foco em defesa e detecção, difere fundamentalmente do desenvolvimento de jogos estratégicos, onde a Teoria dos Jogos constitui a própria base da mecânica do jogo.

No campo da segurança de software a pesquisa intitulada ***"Detecting Misuse os Security APIs: A Systematic Review"*** Mousavi *et al.* (2025) analisam como o uso inadequado de APIs de segurança podem gerar vulnerabilidades sérias em sistemas, particularmente em áreas como criptografia, autenticação e integridade dos dados. Os autores destacam que a complexidade dessas APIs, combinada com a falta de documentação apropriada, é um dos principais motivos que fazer com que os desenvolvedores a cometerem erros, levando a falhas de segurança, exposição indevida de dados e dificuldade no cumprimento de normas regulatórias. Embora o estudo não utilize diretamente modelos da Teoria dos Jogos, sua relevância para esta seção se deve à análise de cenários onde agentes adversários exploram decisões técnicas mal formuladas, evidenciando a natureza estratégica das interações entre atacantes e defensores.

A principal contribuição desse estudo para o presente trabalho está na ênfase dada à importância de uma documentação clara, precisa e que atenda às exigências dos desenvolvedores. Ao mostrar como especificações que são incompletas ou mal elaboradas aumentam consideravelmente o risco de uso incorreto, o estudo reforça o papel central da documentação como elemento de redução como elemento de reduções de riscos, um conceito que se conecta diretamente ao objetivo desta pesquisa que propõe uma API voltada para jogos de estratégia, a qual, além de modelar as interações entre agentes, prioriza uma documentação que visa reduzir ambiguidades e guiar corretamente o desenvolvimento de agentes inteligentes.

3.3 Teoria dos Jogos em Disciplinas Específicas da Engenharia de Software

Esta seção analisa trabalhos que focam na aplicação da Teoria dos Jogos em áreas muito específicas e técnicas do desenvolvimento de software, como a verificação e a definição de requisitos. Esses estudos evidenciam o detalhamento e a especificidade com as quais a teoria pode ser aplicada.

No artigo ***"Implementation of Game Theory in Software Testing"*** Wukkadada e Pendbhaje (2018) analisaram o processo de desenvolvimento de software sob a ótica de um jogo não cooperativo. Eles investigaram como os objetivos individuais de cada participante, que agem em seu próprio interesse, podem gerar disputas que afetam o sucesso do projeto.

A relação com o presente projeto reside na aplicação da Teoria dos Jogos em um contexto de Engenharia de Software para estudar interações complexas. Entretanto, o foco deles é na melhoria da qualidade e na avaliação de testes de software, ao passo que esta pesquisa se dedica à modelagem da lógica do jogo em si à documentação de APIs para facilitar a interação com agentes inteligentes, o que representa um objetivo final distinto.

3.4 Gamificação e APIs Educacionais

Nesta seção, são revisados estudos que exploram a aplicação de elementos de jogos em ambientes de educacionais e corporativos, com ênfase em métodos que implementam mecânicas de gamificação por meio de APIs. Esses trabalhos evidenciam como funcionalidades como pontos, conquistas e classificações podem ser simplificados em interfaces que podem ser reaproveitadas e aplicadas de maneira uniforme em diferentes contextos.

A literatura sobre gamificação destaca como os elementos de jogos (Como pontos,

conquistas, classificações e recompensas) podem ser incorporados a ambientes não relacionados ao jogo para incrementar a motivação e o envolvimento. O trabalho de Boas *et al.* (2017) ***"GAMAPI - Uma API para Gamificação"***, oferece um exemplo prático dessa combinação ao sugerir uma API destinada à aplicação de mecanismos de gamificação em ambientes educacionais e de treinamento. Sua abordagem se fundamenta na disponibilização de serviços centralizados para gerenciar pontuações, conquistas e placares, ressaltando a relevância de uma interface clara e de um processamento que ocorra completamente no servidor por meio de *Web Services*. Ao analisar a estrutura desta API, a pesquisa destaca como a simplificação das mecânicas de jogo em uma interface coesa pode facilitar sua implementação em diversos contexto.

A contribuição principal deste trabalho para o estudo atual se reside na demonstração de como as mecânicas de jogos podem ser organizadas em uma API que é reutilizável e documentada de maneira compreensível. Ao propor uma solução modular para a gamificação estrutural API desenvolvida neste trabalho, enfatizando a necessidade de *endpoints* bem definidos, regras coerentes e documentação detalhada para facilitar o uso correto por desenvolvedores externos e agentes inteligentes.

3.5 Frameworks e APIs para Jogos Estratégicos

Esta seção apresenta pesquisas voltadas à criação de *frameworks* e APIs que estruturam jogos de estratégia como plataformas amplas para testes e desenvolvimento de IA. Os estudos enfatizam a importância da padronização de estados, ações e regras, possibilitando que agentes e algoritmos sejam avaliados em cenários variados e comparáveis.

Pesquisas voltadas à criação de *frameworks* e APIs para jogos estratégicos têm desempenhado papel importante no progresso da IA em jogos. O trabalho de Dockhorn *et al.* (2020) ***"STRATEGA - A General Strategy Games Framework"*** exemplifica essa direção ao fornecer uma plataforma que é capaz de unificar diferentes jogos de estratégia sob um conjunto comum de interfaces, possibilitando que algoritmos de *Inteligência Artificial* (IA) sejam testados em vários cenários sem depender de um jogo específico. Ao apresentar uma API que padroniza regras, ações, estados e interações, a pesquisa enfatiza as vantagens da generalização, minimizando vieses e ajustes excessivos a ambientes, permitindo a avaliação de agentes em múltiplos contextos e tornando as comparações entre métodos mais simples.

A relevância deste trabalho para a pesquisa em questão é notável, pois ilustra como os jogos estratégicos podem ser representados em uma API que descreve estados ações e interações

de maneira transparente. Os fundamentos aplicados no STRATEGA sublinham a necessidade de criar uma interface que seja consistente e previsível para agentes inteligentes, um conceito central deste estudo, que visa documentar uma API centrada na lógica interna de jogos de estratégia.

No trabalho de Silva (2015) ***"Inteligência Artificial Adaptativa para Ajuste Dinâmico de Dificuldade em Jogos Digitais"***, o Ajuste Dinâmico de Dificuldade (DDA) refere-se a um método que visa modificar o grau de dificuldade em jogos digitais de acorde com o desempenho do jogador, proporcionando um melhor equilíbrio entre desafio e habilidade. Pesquisas na área mostram que os níveis de dificuldade fixos ´podem resultar em frustração quando são muito altos ou em desinteresse quando são muito baixos. A inteligência artificial adaptativa, portanto aparece como uma solução para ajustar continuamente e de forma responsiva parâmetros, comportamentos e até aspectos do design de níveis de maneira personalizada.

A relevância deste campo de pesquisa para o estudo atual reside, em grande parte, na conexão entre modelagem de comportamento, técnicas de jogo e a necessidade de definir claramente as regras que regulam a interação entre diferentes agentes. Compreender essas abordagens é fundamental para criar APIs que representem de maneira clara estados, ações e suas consequências, possibilitando o desenvolvimento de agentes inteligentes que considerem dinâmicas adaptativas e raciocínio estratégico.

3.6 Documentação de API e Aprendizado de Desenvolvedores

Nesta seção são apresentados estudos que exploram a maneira como desenvolvedores aprendem e utilizam a documentação de APIs, reconhecendo práticas, desafios e necessidades frequentes. Essas pesquisa ressaltam a relevância de fornecer exemplos nítidos, uma estrutura clara e um contexto apropriado para promover a adoção e prevenir o uso incorreta de interfaces de programação.

O estudo ***"How developers use API documentation: an observation study"*** Meng, Steinhardt e Schubert (2019), examina a maneira como os desenvolvedores interagem com a documentação ao se familiarizarem com uma nova API, identificando dificuldades comuns ligadas à organização, clareza e disponibilidade de exemplos práticos. Os autores observam que os desenvolvedores buscam rapidamente um resumo sobre as funcionalidades, pontos de acesso e exemplos tangíveis que demonstrem a aplicação real da API. Questões como a documentação inconsistente, falta de contexto ou ausência de cenários ilustrativos afetam diretamente a velocidade de aprendizado e podem resultar em um uso inadequado das funcionalidades. A pesquisa

destaca a relevância de garantir que a documentação esteja em harmonia com as expectativas e métodos cognitivos que os programadores utilizam ao explorar uma nova interface.

A relevância dente estudo para a pesquisa atual é clara e essencial, ele enfatiza a importância de uma documentação bem estruturada, clara e voltada para aplicações práticas, especialmente importante em uma API destinada a jogos estratégicos, que será utilizada por desenvolvedores e agentes inteligentes. As diretrizes observadas nesta pesquisa influenciam a forma como a documentação será organizada e redigida neste trabalho, que busca oferecer descrições diretas, exemplos práticos e explicações que auxiliem na implementação correta e evitem ambiguidades que poderiam comprometer o desempenho dos agentes.

3.7 Conclusão da Revisão de Literatura

De maneira geral, os trabalhos revisados evidenciam a crescente relevância e adaptabilidade da Teoria dos Jogos em diversas áreas da Engenharia de Software. Eles ressaltam a capacidade dessa teoria em modelar interações complexas, antecipar resultados e otimizar procedimentos. Embora cada pesquisa aborde a Teoria dos Jogos de uma maneira única, seja na gestão de requisitos, no desenvolvimento de estratégias de defesa, na verificação de software ou em cenários competitivos, todos ilustram como modelos estratégicos podem facilitar a compreensão das interações organizadas entre diferentes agentes.

Além disso, as pesquisas sobre gamificação de APIs educacionais reafirmam a importância de organizar mecânicas de jogos em interfaces que posam ser reutilizadas mostrando como aspectos lúdicos podem ser traduzidos em componentes técnicos claros e bem definidos. Da mesma forma investigações sobre estruturas de jogos estratégicos e inteligência artificial adaptativa enfatizam a necessidade de representações formais de estados, ações e regras de jogo, que são essenciais para a criação de agentes inteligentes que consigam raciocinar estrategicamente e ajustar seu comportamento. Esses estudos evidenciam que, em ambientes interativos e em constante mudança, a clareza estruturar do sistema é tão crucial quanto o próprio modelo estratégico.

Os estudos que analisam a utilização e o uso inadequado de APIs, principalmente em contextos de segurança sublinham ainda mais a necessidade de uma documentação clara, exemplos apropriados e diretrizes específicas. A literatura indica que falhas nessa documentação podem resultar em vulnerabilidades, interpretações errôneas e comportamentos imprevisíveis, enfatizando a necessidade de interfaces que sejam projetadas não apenas para operar mas também

para serem compreendidas e usadas corretamente. Essa questão se relaciona diretamente com estudos sobre como desenvolvedores assimilam novas APIs, ressaltando a importância de arranjos que ajudem na descoberta de funcionalidades, identifiquem pontos de entrada e incentivem um aprendizado gradual.

Assim, a implementação de uma metodologia voltada para a documentação de APIs de jogos estratégicos, fundamentada na Teoria dos Jogos representa uma área com significativo potencial para a exploração. Essa abordagem se apoia nas melhores práticas das áreas analisadas, incluindo modelagem estratégicas, análise de incentivos e boas práticas de documentação e abstração de mecânicas de jogos. Ao unir esses conceitos, o presente estudo sugere uma documentação que possa explicitar de maneira clara a lógica interna dos jogos estratégicos, oferecer suporte adequado ao desenvolvimento de agentes inteligentes e evitar ambiguidade que possam prejudicar a operação da API. Dessa forma, contribui para preencher uma lacuna ainda pouco abordada na literatura, criando uma base estruturada para investigações futuras e para aplicações práticas em jogos estratégicos e IA.

4 METODOLOGIA

A metodologia adotada neste trabalho estabelecer um processo estruturado e replicável para a documentação de APIs aplicadas a jogos de estratégia. O conteúdo da proposta é fundamentado na interseção entre os princípios da Engenharia de Software, a Arquitetura de Software (com foco na modularização), a documentação de sistemas e os conceitos da Teoria dos Jogos, utilizados como referencial conceitual para organização de agentes, estados e decisões.

O foco da metodologia está na modelagem conceitual e na documentação da API, com o objetivo de promover clareza, organização, reutilização e extensibilidade da estrutura proposta para outros jogos de estratégias com características semelhantes. è importante destacar que este trabalho não abrange a implementação da API ou a realização de experimentos computacionais, restringindo-se à definição metodológica e à documentação técnica.

4.1 Estrutura Metodológica

A Metodologia foi estruturada em seis fases sequenciais, cada uma ligada aos objetivos específicos do presente trabalho:

- Análise conceitual de jogos de estratégia;
- Levantamento de requisitos funcionais e não funcionais da API;
- Modelagem conceitual e arquitetural da API;
- Elaboração da documentação técnica da API;
- Aplicação da metodologia em um estudo de caso;
- Revisão e aprimoramento com base em *feedback* e boas práticas.

Cada etapa foi fundamentada em referências clássicas e modernas na área como Sommerville (2019), Tonini, Carvalho e Spínola (2008), Robillard e Deline (2011), Francez (2017), Figueiredo (1994), além de conceitos essenciais da Teoria dos Jogos, apresentados em obras como ***"Theory of Games and Economic Behavior"*** (Neumann; Morgenstern, 1944). Os conceitos da Teoria dos Jogos são empregados de forma descritiva e organizacional, sem a aplicação direta de modelos matemáticos ou solução de equilíbrios estratégicos.

4.2 Análise do Jogo e Levantamento de Requisitos

A etapa inicial consistiu em uma análise conceitual *Clue Suspeitos*, com o objetivo de compreender seus principais elementos estruturais, tais como agentes, estados do jogo, decisões

possíveis e regras gerais. Essa análise teve como finalidade contribuir com o levantamento de requisitos da API, e não modelar integralmente a lógica do jogo.

Foram identificados requisitos funcionais relacionados às ações disponíveis, às verificações de estados e às interações possíveis entre os agentes, assim como requisitos não funcionais associados à modularidade, clareza, organização e possibilidade de reutilização da estrutura documentada. A avaliação considerou jogos estratégicos como contextos de decisão com dados incompletos conforme abordado por Figueiredo (1994).

Embora o *Clue Suspeitos* tenha sido utilizado como estudo de caso, a metodologia apresentada não se restringe a esse jogo particular. A estratégia utilizada permite sua utilização em outros jogos de estratégia que apresentem vários agentes e decisões baseadas em estados do sistema, permanecendo em conformidade com a estrutura teórica estabelecida.

4.3 Modelagem Arquitetural da API

Com base nos requistos levantados, foi feita a elaboração da arquitetura da API. Nesta etapa, foram utilizados diagramas da *Unified Modeling Language* (UML), incluindo diagramas de casos e uso, sequência e classes, com o objetivo de representar as funcionalidades e atribuições dos componentes da API.

Além disso, foi utilizado o modelo C4 para mostrar a arquitetura em diversos níveis de abstração (Contexto, *Containers* e Componentes), possibilitando uma visão gradual da estrutura do sistema. É importante notar que o nível de Código foi tratado apenas de maneira conceitual, uma vez que não se incluiu implementação no escopo do projeto.

Os elementos do jogo foram convertidos em estruturas da API, onde as ações são vistas como operações, os agentes como entidades conceituais e os estados do jogo como dados mantidos e acessados pelo sistema. Esta metodologia segue as orientações de Meng, Steinhardt e Schubert (2018), que enfatizam a relevância de representações visuais e descrições conceituais na documentação de APIs.

4.4 Documentação Técnica e Padrões Utilizados

A documentação da API foi elaborada com base em boas práticas descritas por Cummaudo *et al.* (2022) e Araujo e Chagas (2023). Foram utilizados diagramas UML e C4, descrições funcionais, tabelas de parâmetros, exemplos de uso e fluxos de interação, com uma organização

modular.

Cada módulo documentado detalha seu objetivo, responsabilidades, entradas e saídas previstas, além de apresentar exemplos que ilustram seu uso. A ênfase na clareza, padronização e na eliminação de ambiguidades seguiu as orientações propostas por Khan *et al.* (2021), bem como as observações de Robillard e Deline (2011) sobre os efeitos prejudiciais da documentação deficiente na manutenção de sistemas.

4.5 Aplicação da Metodologia no Estudo de Caso

A metodologia proposta foi aplicada ao estudo de caso do jogo *Clue Suspeitos* com o objetivo de demonstrar sua aplicabilidade, ao invés de validar a API por meio de testes experimentais. O processo envolveu a criação de toda a documentação da API, levando em consideração distintos cenários de uso, perfis de jogadores e variações no conceito do jogo.

A análise da documentação criada levou em conta aspectos qualitativos, como clareza, uniformidade, estrutura e possibilidade de reaproveitamento. Essa revisão teve um caráter descritivo e investigativo, fundamentada em diretrizes para análise de documentação técnica apresentadas por Khan *et al.* (2021) e Taylor *et al.* (2019).

Essa etapa permitiu perceber de que maneira a metodologia pode ser utilizada para organizar a documentação de APIs direcionadas a jogos de estratégia, destacando sua adaptabilidade e relevância para sistemas que envolvem múltiplos agentes e decisões estratégicas.

4.6 Integração entre Engenharia de Software e a Teoria dos Jogos

A etapa final envolveu o exame da conexão teórica entre Engenharia de Software e Teoria dos Jogos dentro do contexto da metodologia apresentada. Os princípios da Teoria dos Jogos, incluindo agentes racionais, escolhas estratégicas e interações em situações de informação incompleta, forma empregadas como fundamentos para estruturar os componentes da API, sem utilizar diretamente modelos como o Equilíbrio de Nash.

A Engenharia de Software trouxe à tona ideias sobre modularização, divisão de responsabilidades e rastreabilidade, conforme mencionado por Sommerville (2019), assegurando uma estrutura clara e bem organizada. Dessa maneira, a metodologia sugerida procura integrar conceitos estratégicos e práticas estabelecidas da engenharia, resultando em uma abordagem apropriada para a documentação de APIs focadas em jogos de estratégia.

5 RESULTADOS

Os resultados obtidos pela aplicação da metodologia sugerida possibilitaram a criação de um conjunto sólido de modelos que retratam, de maneira clara e organizada, a estrutura da API para jogos de estratégia. Tomando o jogo *Clue Suspeitos* como exemplo, foram elaborados diagrama de caso e uso, diagrama de classe e diagramas de sequência que mostram tanto a lógica interna do software quanto o fluxo de interações entre os jogadores e o sistema. Apesar de *Clue Suspeitos* ter servido como base para a modelagem, a estrutura gerada foi de forma intencional genérica, podendo atuar como referencial para a criação de APIs para uma variedade de jogos com características estratégicas.

A seguir cada resultado será apresentado e examinado de maneira minuciosa, evidenciando a eficácia da metodologia ao converter conceitos abstratos, como agentes, ações estratégias e estados em artefatos formais capazes de guiar a documentação automatizadas realizadas por agentes inteligentes.

5.1 Diagrama de Caso e Uso: Representação das Interações Essenciais

O diagrama de caso e uso elaborado oferece uma representação clara e direta das principais atividade que um jogador pode realizar no sistema (Figura 1). Este modelo é essencial, pois retrata a visão externa da API, destacando as funcionalidades que precisam estar acessíveis ao usuário e definindo demarcações claras entre a lógica do jogo e as responsabilidades do jogador ou de um agente inteligente.

Os quatro casos de uso identificados, Iniciar Partida, Realizar Pergunta, Responder Pergunta e Fazer Acusação, refletem diretamente a dinâmica estratégica do jogo:

- **Iniciar Partida:** se refere ao momento que o sistema configura o ambiente e determina o conjunto inicial de agentes e cartas;
- **Realizar Pergunta:** materializa a essência do jogo de dedução, permitindo que o jogador obtenha informações parciais;
- **Responder Pergunta:** descreve o fluxo oposto, no qual outo jogador deve revelar ou rejeitar informações estratégicas;
- **Fazer Acusação:** representa a jogada final, de caráter decisivo, que influencia diretamente o desfecho da partida.

Este diagrama ressalta que a API deve incluir interações fundamentais da estrutura lógica

do jogo, assegurando que agentes externos, sejam eles humanos ou artificiais tenham aceso a todas as operações necessárias para participar de maneira eficaz do fluxo estratégico.

Além disso, a modelagem demonstra que a API está focada no jogador, que é o responsável por iniciar todas as interações pertinentes. Essa estrutura se alinha aos *frameworks* de jogos estratégicos e às diretrizes de usabilidade para APIs, nas quais operações devem ser iniciadas por entidades externas, e não pelo sistema por conta própria.

Figura 1 – Diagrama de Caso e Uso.

Fonte: Autoria Própria(2025)

5.2 Modelagem Estrutural: A Abordagem Hierárquica C4 (Níveis 1, 2, 3 e 4)

A arquitetura elaborada nesse projeto orientou-se pelo modelo C4, que ordena a arquitetura de software em quatro níveis hierárquicos de abstração: Contexto, Containers, Componentes e Código. A utilização desse modelo permitiu representar o sistema de forma gradual, desde uma visão abrangente das interações externas até os os detalhes internos da implementação. Tal abordagem foi fundamental para transformar a lógica estratégica do jogo *Clue Suspeitos* em uma estrutura clara, modular expansível e que esteja alinhada com os princípios estabelecidos da Engenharia de Software, como modularidade, baixo acoplamento, separação de responsabilidades e rastreabilidade.

Além disso, o modelo C4 se mostrou especialmente adequado para este projeto, pois possibilitou a inclusão direta de conceitos da Teoria dos Jogos, como agentes, ações, estratégias, estados e informações incompletas, na arquitetura, fortalecendo a coerência entre o comportamento estratégico modelado e sua representação computacional.

A seguir, são apresentados os quatro níveis de modelagem, cada um acompanhado dos diagramas correspondentes produzidos.

5.2.1 Nível 1: Diagrama de Contexto

O diagrama de Contexto apresenta uma visão geral do **Sistema de Jogo Clue Suspeitos**, sublinhando suas fronteiras e interações com sistemas externos. Nesse nível, o sistema é mostrado como um único bloco funcional, ocultando sua complexidade interna e enfatizando as relações de dependência com outros serviços.

No modelo criado (figura 2), o **Sistema de Jogo Clue Suspeitos** estabelece uma conexão direta com um **Serviço de Autenticação e Perfis**, responsável pelo login, registro e identificação dos usuários, bem como o **Jogador**, que atua como agente externo responsável por iniciar todas as interações estratégicas com o sistema. Essa escolha arquitetural reforça o princípio da separação de preocupações ao atribuir a autenticação a um sistema específico, permitindo que a API se concentre apenas na lógica estratégica do jogo.

O **Jogador** é representado no diagrama como uma entidade que não faz parte do sistema, podendo ser um usuário humano ou um agente inteligente, que se conecta à API através de uma interface. Essa representação está de acordo com os princípios da Teoria dos Jogos, onde os agentes fazem escolhas com base nas informações que têm e impactam a situação geral do jogo por meio de suas decisões.

As principais contribuições deste nível:

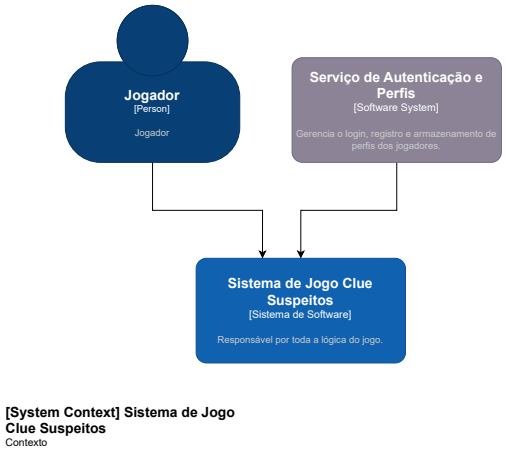
- Define os limites do sistema, deixando claro que toda a lógica do jogo está contida na API;
- Identifica as dependências externas, promovendo o isolamento e a modularidade;
- Prepara o sistema para futuras integrações com plataformas de jogos reais ou serviços de identidade;
- Reflete a dinâmica da Teoria dos Jogos ao identificar jogadores como agentes externos (jogadores) que interagem estrategicamente com o sistema.

5.2.2 Nível 2: Diagrama de *Containers*

O diagrama de *containers* oferece uma visão mais detalhada do que foi apresentado no nível anterior, descrevendo os blocos executáveis que formam a aplicação. No modelo elaborado (figura 3), o sistema foi segmentado em três *containers* principais:

- **API de Jogo Estratégico:** é o *container* responsável por toda a lógica do jogo, pelo

Figura 2 – Diagrama de Contexto.



Fonte: Autoria Própria(2025)

gerenciamento dos estados das partidas, pela aplicação das regras, pela mediação das interações entre os jogadores e pela comunicação com o banco de dados;

- **Banco de Dados do Jogo:** é o *container* responsável por armazenar os dados persistentes, informações estáticas, dados dos jogadores e configurações gerais do jogo.
- **Interface Gráfica:** é o *container* que gerencia a interface visual do sistema, através do qual os usuários se envolvem como o jogo. Esse *container* coleta as ações dos usuários e envia as demandas ao controlador da aplicação, sem incluir lógicaa estratégica ou normas de opeçãoo.

Os principais fluxos que foram representados:

- A interface gráfica envia as requisições realizadas pelos jogadores ao controlador da aplicação, que delega o processamento à API do jogo Estratégico;
- A API realiza a leitura e gravações de dados no banco de dados continuamente, garantindo a consistência na evolução do jogo;
- Todas as instruções, como acusar, questionar, responder e iniciar jogos, são originadas pelo jogador e processadas pela API, que valida e implementa a lógica necessária.

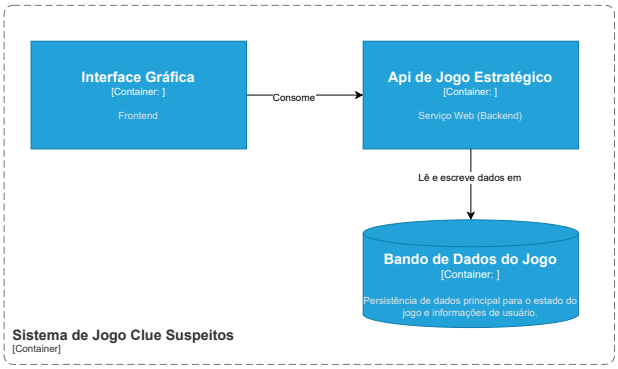
A principal contribuição deste nível, é que ele demonstra que a metodologia sugerida pode ser aplicada de forma mais ampla: qualquer jogo que se baseie em estratégias, agentes e estados pode empregar a mesma estrutura básica, uma API central e um repositório persistente. Além disso:

- O modelo promove boas praticas em arquitetura orientada a serviços;
- Assegura a API permaneça desacoplada da camada de armazenamento, permitindo futuras

substituições;

- Proporciona escalabilidade horizontal ao dividir lógica de dados.

Figura 3 – Diagrama de *Container*.



Fonte: Autoria Própria(2025)

5.2.3 Nível 3: Diagrama de Componentes

O diagrama de componentes oferece uma visão mais detalhada da API, quebrando sua lógica interna em módulos especializados. Esta é a camada que mais ressalta a modularização do sistema e mostra como as responsabilidades foram alocadas para aumentar a clareza, a capacidade de expansão e a coerência.

O modelo apresentado (figura 4) inclui cinco módulos principais:

- **Módulo de Partida:** administra o clico de vida das partidas, turnos e estados gerais;
- **Módulos de Sugestões:** gerencia perguntas e respostas entre os jogadores, servindo como o núcleo da manipulação de informações incompletas.
- **módulo de Acusações:** concentra a lógica das acusações finais e a validação da solução secreta.
- **Módulo de Dados do Jogo:** media o acesso ao banco de dados, assegurando abstração e persistência adequada;
- **Módulo de Regras:** implementa as regras do jogo e assegura a coerência lógica em todas as ações.

Tendo os seguintes aspectos incorporados:

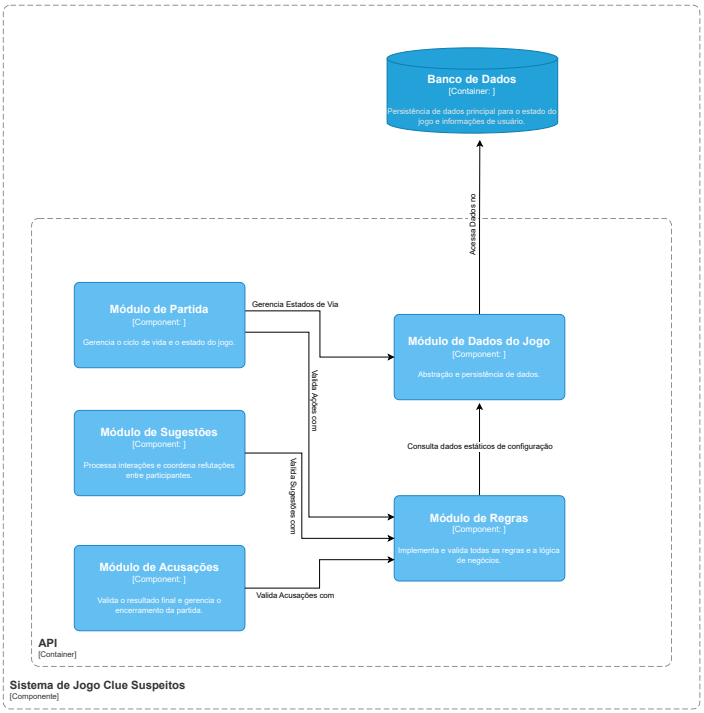
- Cada módulo representa um conjunto de ações estratégicas, refletindo diretamente conceitos da Teoria dos Jogos;

- Informações privadas, públicas e parciais são organizadas de maneira clara entre os componentes;
- A distinção entre regras e estado fortalece a capacidade de evolução e reutilização.

A principal contribuição deste nível, reforça a perspectiva modular necessária para uma documentação da API que seja clara e precisa. Além disso:

- Auxilia na formação de *endpoints* organizados por áreas de atuação;
- Torna a leitura mais acessível para agentes inteligentes que utilizam a API;
- Incorpora rigor conceitual ao mapeamento das interações lógicas dentro do jogo.

Figura 4 – Diagrama de Componentes.



Fonte: Autoria Própria(2025)

5.2.4 Nível 4: Código - Diagrama de Classe

O nível de código descreve a estrutura interna do sistema, mostrando classes, características, funções e conexões. Neste estudo, o modelo de código foi ilustrado pelo diagrama de classe (figura 5), que traduz diretamente a lógica estratégica do jogo em entidades computacionais formais.

É neste ponto que o método sugerido evidência sua eficácia: as abstrações fundamentais da Teoria dos Jogos, como estados, ações, agentes, incertezas e resultados, são apresentadas de

maneira clara em classes que são manipuladas pela API.

O sistema foi estruturado em classe que representam diretamente os componentes do jogo:

- **Menu do Jogo:** o ponto inicial da API responsável por coordenar as solicitações externas e gerenciar as partidas em andamento;
- **Partida:** classe que sintetiza o estado geral de sessão de jogo, incluindo informações sobre jogadores, cartas distribuídas e a solução secreta;
- **Jogador:** simboliza cada agente participante, incluindo cartas pessoais, nome, estados e ações disponíveis;
- **Pergunta:** estrutura destinada a registar as perguntas feitas, quem fez, quem recebeu e as cartas que foram reveladas;
- **Carta/TipoCarta:** abstraem os elementos principais da mecânica, relacionados aos eixos (suspeitos, arma, local);
- **Confidencial:** conjunto oculto de cartas que formam a solução para o mistério.

Cada classe contém atributos e métodos que refletem diretamente o comportamento esperado durante o jogo, assegurando que haja coerência entre a lógica estratégica documentada e as estruturas formais.

O diagrama evidencia a aplicação de princípios fundamentais da Engenharia de Software:

- Alta coesão: cada classe tem uma responsabilidade bem definida e específica;
- Baixo acoplamento: os módulos interagem através de interfaces bem definidas, diminuindo dependências indesejadas;
- Modularização arquitetural: separação entre a lógica da partida, a lógica do jogador, regras confidenciais e interação externa;
- Representação explícita de agentes e ações: reforçando a relevância da Teoria dos Jogos.

A classe Partida, por exemplo, centraliza a lógica para a verificação das acusações, enquanto a classe Jogador concentra atividades relacionadas a perguntas, respostas e posse de cartas, refletindo a real divisão de responsabilidades no jogo.

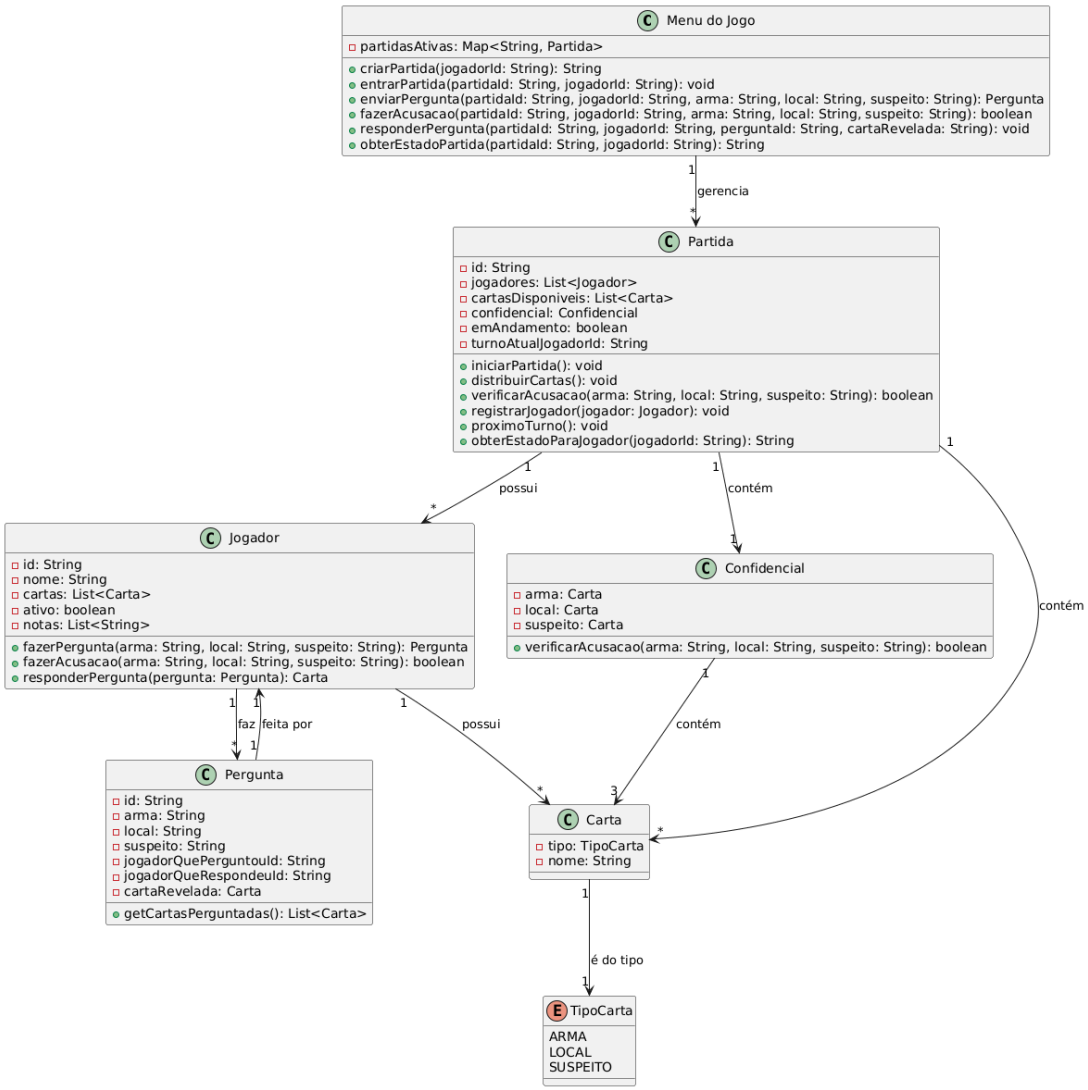
Embora seja baseado no jogo *Clue Suspeitos*, o diagrama possibilita a expansão para outros jogos de estratégia que apresentam características similares como:

- Múltiplos agentes;
- Informações incompletas ou privadas;
- Ações sequenciais;

- Estados persistentes;
- Eventos dependentes de regras;
- Interações entre jogadores.

Dessa forma, o resultado dessa modelagem serve como uma estrutura conceitual para APIs destinadas a jogos de estratégia.

Figura 5 – Diagrama de Classe.



Fonte: Autoria Própria(2025)

5.3 Diagramas de Sequência: Dinâmica Temporal das Interações Estratégicas

Os diagramas de sequência são fundamentais para ilustrar como as informações se movem, a ordem dos eventos e a participação das classes durante as ações realizadas no jogo.

Quatro fluxos principais foram modelados.

5.3.1 Diagrama de Sequência: Iniciar Partida

Este Diagrama ilustra o procedimento de criação e início de uma nova partida. Destaca:

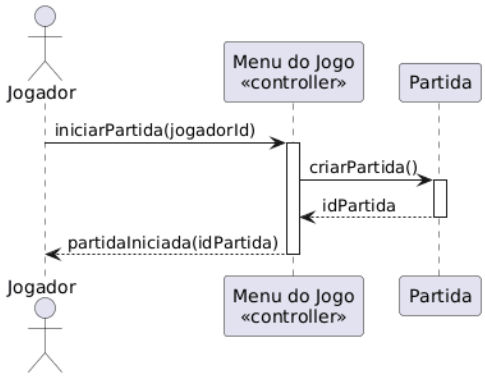
- A requisição do jogador;
- A formação de uma nova instância de partida;
- O retorno de um identificador exclusivo.

Esse fluxo assegura que a API mantenha um controle central sobre a gestão dos estados, prevenindo conflitos, duplicações ou sobreposições de partidas. Além disso confirma a necessidade de:

- Manutenção de estados;
- Gerenciamento de várias partidas ao mesmo tempo;
- Isolamento entre sessões;

que são elementos essenciais em jogos com múltiplos agentes.

Figura 6 – Diagrama de Sequência Iniciar Partida.



Fonte: Autoria Própria(2025)

5.3.2 Diagrama de Sequência: Realizar Pergunta

O diagrama de sequência referente a essa ação é o mais complexo, pois reflete a mecânica central do jogo: a coleta de informações confidenciais.

Ele demonstra:

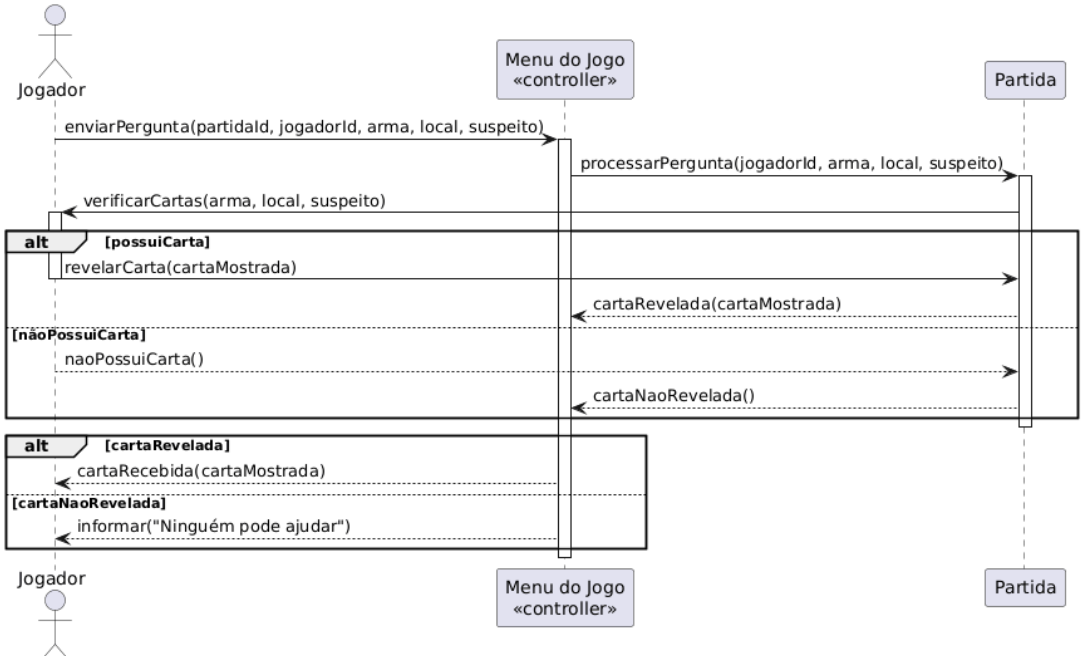
- A formulação da pergunta pelo jogador;
- O envio ao Menu do Jogo;
- A verificação das cartas na classe Partida;
- A opção de outro jogador revelar ou não uma carta,
- Os dois resultados possíveis: carta revelada ou a falta de informações.

Esse modelo captura a essência da Teoria dos Jogos adaptada ao contexto:

- Informação incompleta;
- Estratégias condicionais;
- Decisões influenciadas pelo comportamento de outros agentes;
- Efeito direto na utilidade e percepção dos jogadores.

Assim, a API não apenas representa as mecânicas, mas também apoia a formação de raciocínios estratégicos.

Figura 7 – Diagrama de Sequência Realizar Pergunta.



Fonte: Autoria Própria(2025)

5.3.3 Diagrama de Sequência: Responder Pergunta

O diagrama de sequência referente ao fluxo de Responder Pergunta é umas das interações mais importantes do jogo, pois envolve um controle cuidadoso na troca de informações entre

dois jogadores. Esse diagrama mostra como uma API lida com a lógica de resposta quando um jogador é escolhido para decidir se deve ou não mostrar uma carta relacionada a uma pergunta feita por outro jogador.

Ele descreve:

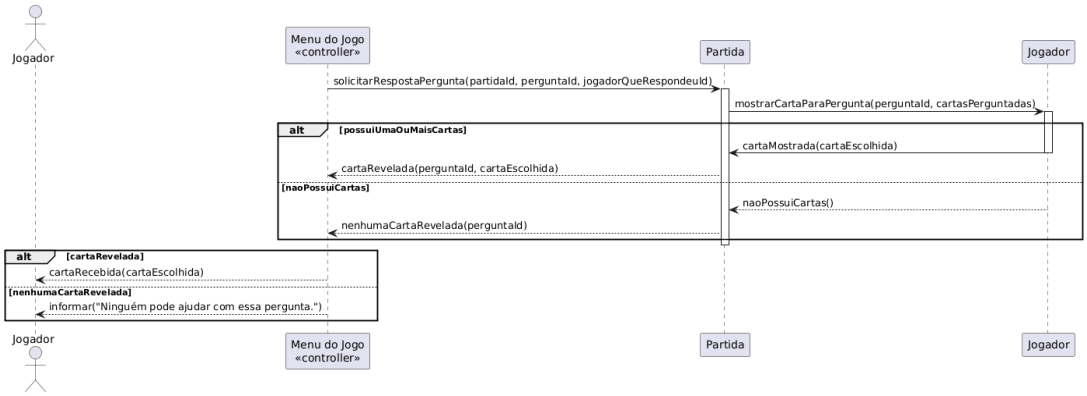
- A solicitação feita pelo Menu do Jogo para captar a resposta do jogador certo;
- A análise das cartas disponíveis para a resposta realizada pela classe Partida;
- A interação direta com o jogador que precisa responder;
- A interação direta com o jogador que precisa responder;
- As opções disponíveis: apresentação de uma carta compatível ou a falta de cartas correspondentes;
- O retorno das informações ao jogador que fez a pergunta.

Assim como nos outros fluxos, esse diagrama destaca mecanismos fundamentais para o jogo de dedução, especialmente no que se refere à gestão de informações incompletas e à importância de preservar a parcialidade dos dados revelados. Ele reforça ideias da Teoria dos Jogos, tais como:

- A desigualdade de informação;
- Decisões baseadas na situação do jogador;
- Efeito estratégico da revelação mínima;
- Manutenção da incerteza como uma parte fundamental do jogo.

Assim, o diagrama ilustra como a API apoia o processo de troca seletiva de informações, essencial tanto para a mecânica do jogo quanto para a capacidade de agentes inteligentes utilizarem estratégias fundamentadas em observação e dedução.

Figura 8 – Diagrama de Sequência Responder Pergunta.



Fonte: Autoria Própria(2025)

5.3.4 Diagrama de Sequência: Fazer Acusação

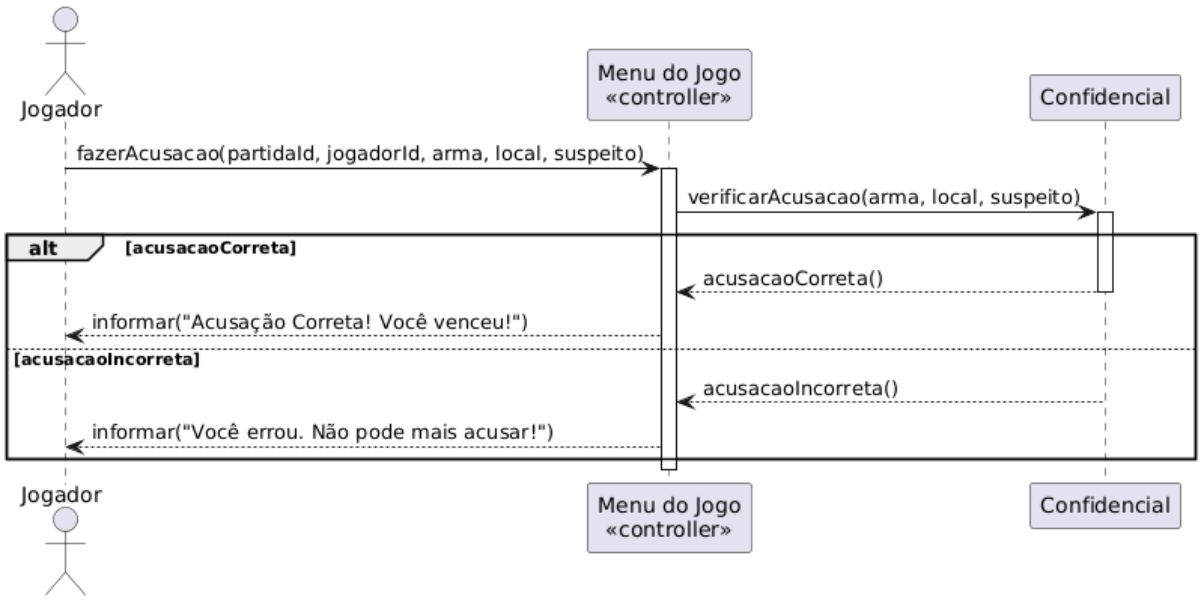
A acusação representa o clímax do jogo. O diagrama criado revela:

- A solicitação inicial do jogador;
- O processamento realizado pelo Menu do Jogo;
- A delegação da verificação para a classe Confidencial;
- O envio da resposta ao jogador.

Ao exemplificar tanto a acusação correta quanto a incorreta, o diagrama esclarece como a API deve administrar resultados que são mutuamente exclusivos e definitivos. Esse modelo:

- Aumenta a transparência no processo;
- Especifica os pontos em que decisões estratégias são tomadas;
- Possibilita que agentes inteligentes reconheçam as consequências de suas ações.

Figura 9 – Diagrama de Sequência Fazer Acusação.



Fonte: Autoria Própria(2025)

5.4 Conclusão dos Resultados

Os resultados indicam que a metodologia aplicada foi capaz de transformar conceitos abstratos da Teoria dos Jogos e da Engenharia de Software em artefatos formais que são coerentes e bem estruturados. A modelagem conta com aspectos importantes, como organização clara,

consistência conceitual e possibilidade de reaproveitamento.

Por outro lado, as declarações sobre a qualidade, eficiência e robustez da abordagem devem ser vistas considerando o foco deste estudo, que está voltado para a documentação e modelagem conceitual, não para a implementação ou avaliação prática de sistemas em uso.

Nesse contexto, os diagramas gerados demonstram o potencial da metodologia como suporte ao desenvolvimento de jogos estratégicos, proporcionando uma base estruturada para documentação, compreensão e futura implementação. Avaliações adicionais, envolvendo implementação prática e estudos comparativos, seriam necessárias para validar de forma mais aprofundada os impactos diretos nas qualidades do software final.

Assim, é possível concluir que a metodologia é adequada e coerente para a modelagem conceitual de APIs voltadas para jogos de estratégia, atingindo os objetivos propostos neste trabalho e podendo servir como referência para trabalhos futuros.

6 CONCLUSÕES E TRABALHOS FUTUROS

A realização desse trabalho possibilitou a elaboração de uma pesquisa fundamentada sobre como a junção entre Engenharia de Software, Documentação de APIs e Teoria dos Jogos podendo contribuir para a modelagem de sistemas interativos que dependem de estratégias, agentes e fluxos complexos de informação. Ao longo do desenvolvimento da pesquisa, buscou-se demonstrar que é possível converter a lógica estratégica do jogo *Clue Suspeitos* em um conjunto de artefatos formais coerentes, aptos a suportar a implementação futura de uma API completa para jogos de dedução.

Este processo envolveu não apenas uma análise aprofundada das mecânicas do jogo, mas também a criação de diagramas, modelos arquiteturais e representações estruturais que refletem de maneira precisa a dinâmica estratégica original. Assim, as conclusões apresentadas aqui resumem os resultados alcançados e indicam direções possíveis para as pesquisas futuras que possam ampliar e solidificar o escopo da investigação.

O primeiro ponto que aparece da análise é que a metodologia abordada se mostrou útil para organizar e sistematizar conceitos abstratos, transformando-os em elementos mais concretos e aplicáveis ao desenvolvimento de software. A aplicação dos princípios da Engenharia de Software foi fundamental para assegurar clareza, modularidade e divisão de responsabilidades em todos os artefatos gerados.

Notou-se que a utilização de diagramas em particular os de sequência, teve um papel relevante na compreensão da lógica do jogo, pois possibilitou visualizar os fluxos temporais das ações, as decisões possíveis e os diferentes resultados que surgem das interações entre jogadores e sistema. Esses diagramas capturam com precisão a natureza estratégica do jogo, onde a informação é parcial, as decisões são condicionais e os agentes precisam raciocinar com base em evidências incompletas. A estrutura produzida demonstrou que a API não é apenas um conjunto fixo de operações, mas um sistema dinâmico capaz de refletir a complexidade original do ambiente do jogo.

A modelagem arquitetural através do modelo C4, também teve um papel significativo na pesquisa. Ao possibilitar a decomposição do sistema em quatro níveis (contexto, *containers*, componentes e código), o modelo forneceu um caminho claro para entender o software tanto de uma perspectiva ampla quanto de uma perspectiva interna.

No nível de contexto, foi possível definir os limites da aplicação e identificar suas interações externas, destacando a API como o núcleo lógico de toda a mecânica do jogo.No nível

mais detalhado, o de componentes, foi possível organizar o sistema em módulos especializados, cada um representado uma parte distinta da lógica estratégica, como a validação de acusações, a formulação de perguntas, a gestão do estado do jogo e o controle das interações entre os jogadores.

Entretanto, é no nível de código que a integração entre teoria e prática atinge seu ponto mais alto, pois o diagrama de classes ilustra como princípios da Teoria dos Jogos, como agentes, ações, estados e incertezas, podem ser diretamente convertidos em entidades de software. As classes mostradas (Partida, Jogador, Pergunta, Confidencial, Carta e Meu do Jogo), não apenas refletem a configuração interna do jogo, mas também destacam a importância de manter alta coesão e baixo acoplamento entre os módulos.

Desse modo, a modelagem gerada atende ao objetivo de estabelecer uma base para uma API que seja tanto funcional quanto extensível e compreensível, aderindo a boas práticas de desenvolvimento de software. Estas característica reforça a importância do trabalho, pois estabelece um caminho para que os sistemas complexos sejam documentados e desenvolvidos com precisão técnica e clareza em seus conceitos.

Uma análise detalhada dos resultados indica que o trabalho ultrapassa a mera representação visual de componentes. Ele estabelece uma conexão entre conceitos teóricos e sua concretização em estruturas formais. Isso se torna especialmente significativo ao observamos a falta de pesquisas focadas na documentação de APIs relacionadas a jogos de estratégia.

Embora haja estudos sobre documentação de forma geral, desenvolvimento de jogos ou sistemas interativos, poucos estabelecem diretrizes específicas para a criação de APIs que tratam de decisões estratégicas diretamente, informações incompletas e agentes racionais. A pesquisa atual ajuda a preencher esta lacuna ao sugerir uma metodologia estruturada que possa ser aplicada, adaptada e ampliada para jogos semelhantes.

Além disso, a criação dos diagramas de sequências promoveu uma compreensão mais aprofundada das interações ao longo do tempo. A modelagem dos fluxos de acusações, formulação de perguntas, respostas as perguntas e início de partidas permitiu a identificação não apenas do comportamento ideal do sistema, mas também dos pontos críticos onde a API precisa e clara. Essa transparência é fundamental não só para os desenvolvedores, mas também para agentes inteligentes que posam utilizar a API em implementações futuras, visto que esses agentes precisam de informações claras e consistentes para operar com base em raciocínios probabilísticos ou algoritmos de decisão.

Assim sendo, as conclusões demonstram que o trabalho cumpriu sua finalidade ao proporcionar uma base conceitual que pode ser reutilizada, ampliada e implementada em sistemas reais. O estudo demonstra que a abordagem é suficientemente sólida para guiar a criação de APIs voltadas para jogos estratégicos e suficientemente versátil para facilitar extensões e adaptações. Essa dualidade entre rigor e flexibilidade se revela essencial em ambientes de desenvolvimento contemporâneos, onde a capacidade de adaptação e evolução e tão essencial quanto a consistência técnica dos artefatos produzidos.

No que diz respeito aos trabalhos futuros, a recomendação mais importante é a execução total da API projetada. Com os diagramas e a estrutura estabelecidos, o próximo estágio é converter essas representações em um código operacional, utilizando *frameworks* apropriados como *Flask* ou *Django REST* que podem ser empregados como exemplos de suporte à implementação, sem que isso implique que a abordagem seja restrita a APIs Web ou que os jogos necessariamente precisem operar em ambientes online. Essa implementação possibilita a validação prática da arquitetura sugerida e a identificação de elementos da modelagem que poderiam ser aprimorados ou melhorados.A escolha da tecnologia dependeria dos requisitos do projeto, podendo incluir também aplicações locais, híbridas ou distribuídas.

Outro caminho relevante para trabalhos futuros está relacionados à implementação de agentes inteligentes que consigam se comunicar diretamente com API. Esses agentes teriam a capacidade de empregar métodos de Teoria dos Jogos, algoritmos estatísticos ou raciocínio bayesiano para efetuar escolhas. A presença desses agentes possibilitaria a realização de simulações completas e testes que analisassem estratégias ideais, padrões comportamentais e a influência de diferentes níveis de informações na dedução feita pelos jogadores. Essa linha de pesquisa abriria um novo campo para a inteligência artificial aplicada e para a compreensão dos jogos de dedução.

Além disso, a pesquisa pode ser alinhada com a criação de interfaces gráficas ou aplicações web que façam uso da API. tal interface proporcionaria aos usuários uma interação intuitiva com o sistema permitindo a visualização de cartas, perguntas, respostas e acusações em um ambiente acessível. Uma interface desse tipo também facilitaria a avalização do sistema com usuários reais, analisando tanto a usabilidade da API sob diversos padrões de utilização.

Em resumo, o trabalho atual apresenta uma base conceitual que une teoria e práticas de forma criteriosa e rigorosa. Os resultados alcançados mostram que é viável representar formalmente a lógica de jogos estratégicos utilizando técnicas estabelecidas de Engenharia de

Software, assegurando clareza estrutural e consistência conceitual. As oportunidades de extensão reforçam a natureza dinâmica da pesquisa e evidenciam que seu potencial de contribuição ultrapassa o escopo inicial, podendo servir de referência para futuros projetos nas áreas de jogos digitais, agentes inteligentes, documentação técnica e sistemas interativos baseados em decisões estratégicas.

REFERÊNCIAS

AJIENKA, N. An application of game theory in the resolution of software user requirements conflicts. 11 2015.

ALMEIDA, R. M. G. de. Impactos da arquitetura de microsserviÇos na manutenÇÃo de sistemas corporativos. **LUMEN ET VIRTUS**, v. 14, n. 32, p. 83–101, Jan. 2024. Disponível em: https://periodicos.newsciencepubl.com/LEV/article/view/5211.

ARAUJO, A. K. S. d.; CHAGAS, J. F. S. Um estudo sobre boas práticas para documentação de APIs. **Brazilian Journal of Development**, v. 9, n. 5, p. 14632–14660, May 2023. ISSN 2525-8761.

BERNHARDT, G. H. *et al.* A utilização da linguagem de modelagem unificada (uml) na engenharia de software: Uma avaliação da adoção e relevância. In: **Anais do 21º Encontro Científico Cultural Interinstitucional**. [S.l.]: Centro Universitário FAG, 2023. ISSN 1980-7406.

BILDIRICI, F.; CODAL, K. S.; MEDENI, T. D. Using agile story points and game theory together: Better software planning and development in agile software development. **arXiv preprint arXiv:2409.12196**, 2024.

BOAS, J. L. V. *et al.* Gamapi - uma api para gamificação. **Informática na educação: teoria amp; prática**, v. 20, n. 1 jan/abr, abr. 2017. Disponível em: https://seer.ufrgs.br/index.php/InfEducTeoriaPratica/article/view/69917.

CUMMAUDO, A. *et al.* Requirements of api documentation: A case study into computer vision services. **IEEE Transactions on Software Engineering**, Institute of Electrical and Electronics Engineers (IEEE), v. 48, n. 6, p. 2010–2027, jun. 2022. ISSN 2326-3881. Disponível em: http://dx.doi.org/10.1109/TSE.2020.3047088.

DOCKHORN, A. *et al.* Stratega: A general strategy games framework. In: **AIIDE Workshops**. [S.l.: s.n.], 2020. p. 1–7.

FIGUEIREDO, R. S. Teoria dos jogos: conceitos, formalização matemática e aplicação à distribuição de custo conjunto. **Miscellaneous**, v. 1, n. 3, p. 273, 1994.

FRANCEZ, D. J. Uma introdução à teoria dos jogos. **Miscellaneous**, v. 19, n. 2, 2017.

GRIGORYAN, G.; COLLINS, A. Game theory for systems engineering: a survey. **International Journal of System of Systems Engineering**, v. 11, p. 121, 01 2021.

HASBRO. **Clue Card Game Instructions**. 2018. https://instructions.hasbro.com/en-au/instruction/clue-card-game. Acesso em: 15 maio 2025.

KHALID, M. N. A.; AL-KADHIMI, A. A.; SINGH, M. M. Recent developments in game-theory approaches for the detection and defense against advanced persistent threats (apts): A systematic review. **Mathematics**, v. 11, n. 6, 2023. ISSN 2227-7390. Disponível em: https://www.mdpi.com/2227-7390/11/6/1353.

KHAN, J. Y. *et al.* **Automatic Detection of Five API Documentation Smells: Practitioners' Perspectives**. 2021. Disponível em: https://arxiv.org/abs/2102.08486.

LUQUE, L. Coesão e acoplamento em sistemas oo. **JavaMagazine**, v. 77, p. 68–74, 03 2010.

MENG, M.; STEINHARDT, S.; SCHUBERT, A. Application programming interface documentation: What do software developers want? **Journal of Technical Writing and Communication**, v. 48, p. 295–330, 07 2018.

MENG, M.; STEINHARDT, S.; SCHUBERT, A. How developers use api documentation: an observation study. **Communication Design Quarterly Review**, ACM New York, NY, USA, v. 7, n. 2, p. 40–49, 2019.

MOUSAVI, Z. *et al.* Detecting misuse of security apis: A systematic review. **ACM Computing Surveys**, ACM New York, NY, v. 57, n. 12, p. 1–39, 2025.

NERY, D. R.; SILVA, P. C. D. Trabalho de Conclusão de Curso, **Arquitetura de Software Modular e Redução de Acoplamento e seu Impacto na Escabilidade de Projetos de Software**. 2022. Disponível em: https://repositorio.animaeducacao.com.br/items/0f7f3987-73d8-489f-8138-34a246638cf4.

NEUMANN, J. von; MORGENSTERN, O. **Theory of Games and Economic Behavior**. first. [S.l.]: princeton, 1944.

ROBILLARD, M. P.; DELINE, R. A field study of api learning obstacles. **Empirical Softw. Engg.**, Kluwer Academic Publishers, USA, v. 16, n. 6, p. 703–732, dez. 2011. ISSN 1382-3256. Disponível em: https://doi.org/10.1007/s10664-010-9150-8.

ROY, S. *et al.* A survey of game theory as applied to network security. In: **2010 43rd Hawaii International Conference on System Sciences**. [S.l.: s.n.], 2010. p. 1–10.

SARTINI, B. A. *et al.* Uma introdução à teoria dos jogos. **Anais da II Bienal da Sociedade Brasileira de Matemática**, p. 25–29, 2004.

SILVA, M. P. Inteligência artificial adaptativa para ajuste dinâmico de dificuldade em jogos digitais. Universidade Federal de Minas Gerais, 2015.

SOMMERVILLE, I. **Engenharia de software**. [S.l.]: Pearson, 2019.

TAYLOR, M. *et al.* Game theory for computer games design. **Games and Culture**, v. 14, n. 7-8, p. 843–855, 2019. Disponível em: https://doi.org/10.1177/15554127740497.

TONINI, A. C.; CARVALHO, M. M.; SPíNOLA, M. Contribuição dos modelos de qualidade e maturidade na melhoria dos processos de software. **Produçao**, 2008.

VITORIANO, V. A. F.; NETO, M. S.; SPERS, E. E. Escolas do pensamento estratégico: uma contribuição a partir da teoria dos jogos. In: **Simpósio Internacional de Administração e Marketing**. [S.l.: s.n.], 2009.

WUKKADADA, B.; PENDBHAJE, A. Implementation of game theory in software testing. In: **4th - Somaiya International Conference on Technology and Information Management (SICTIM'18)**. [S.l.]: K.J. Somaiya Institute of Management Studies and Research (SIMSR), 2018. p. 36–39. ISSN 2278-0661. PP 36-39.

YAZDANI, S. *et al.* A game theory perspective on requirement-based engineering design. In: ____. [S.l.: s.n.], 2018. p. 901–910. ISBN 978-3-319-62216-3.

YILMAZ, M.; O'CONNOR, R. Maximizing the value of the software development process by game theoretic analysis. **ACM International Conference Proceeding Series**, 06 2010.