



# NBA API

Grupo : Alan Mateus;  
Lucas Rickelme;  
Pedro Henrique Gregório.

# PROBLEMÁTICA

*Devido ao grande volume de dados da NBA, a filtragem eficiente de informações específicas, como o desempenho de jogadores em diferentes temporadas ou jogos, torna-se um desafio. A necessidade de combinar dados de múltiplas tabelas (como jogadores, equipes e estatísticas) para gerar análises precisas e rápidas aumenta a complexidade, exigindo sistemas e processos de consulta otimizados.*



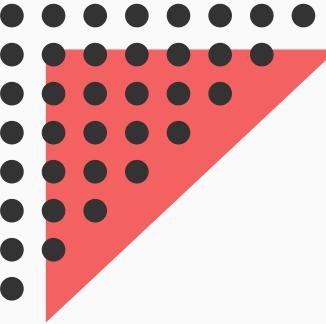
# OBJETIVOS

- Criar e organizar as tabelas do banco de dados da NBA, garantindo a correta integração entre temporadas, equipes, jogadores, jogos e estatísticas.
- Desenvolver uma API com endpoints que possibilitem a inserção manual de dados e interação direta com o banco.
- Implementar técnicas para melhorar o desempenho das consultas, garantindo que a análise de grandes volumes de dados seja realizada de forma rápida e eficiente.

# PÚBLICO ALVO

O público-alvo deste projeto inclui analistas de dados esportivos, gestores de equipes da NBA, desenvolvedores de sistemas esportivos, pesquisadores e estudiosos da NBA, fãs e entusiastas de estatísticas da NBA, e apostadores esportivos, todos os quais buscam acessar, analisar e utilizar dados detalhados para melhorar estratégias, tomar decisões informadas e obter insights sobre o desempenho de jogadores e equipes ao longo das temporadas.





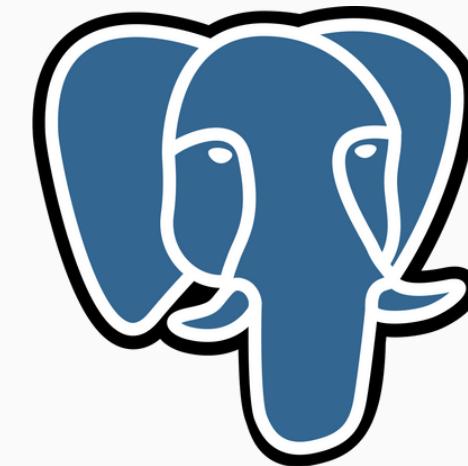
# TECNOLOGIAS UTILIZADAS



JAVA



MAVEN



POSTGRESQL



JUNIT



SPRING BOOT

# DOCUMENTO DE VISÃO



# DOCUMENTO DE VISÃO

O propósito deste documento é expor as necessidades e funcionalidades gerais do sistema, definindo os requisitos de alto nível em termos de necessidades dos usuários finais.



# **ESTRUTURA - DOC DE VISÃO**

- 1. Título**
- 2. Objetivo**
- 3. Escopo**
- 4. Gestor**
- 5. Posicionamento**
- 6. Alternativas e Concorrências**
- 7. Perspectiva do Produto**



# ESTRUTURA - DOC DE VISÃO

## 8. Necessidades e Funcionalidades do Produto

Necessidade 4		Beneficio
Controle de dados estatísticos dos Jogos		<b>Crítico</b>
Id Func.	Descrição das Funcionalidades/atores envolvidos	
F4.1	Listagem de jogos	<p><b>Usuário</b> – Poderá consultar os dados dos jogos através de requisições API via uma interface. A API retornará os dados da tabela "jogos" estruturados em formato JSON.</p> <p>Exemplo de resposta: {data:[{ id: 1, confronto: "Time A vs Time</p>

Necessidade 6		Beneficio
Listagem de estatísticas dos times		<b>Crítico</b>
Id Func.	Descrição das funcionalidades, autores envolvidos e discriminação dos dados necessários	
F6.1	Listagem de estatísticas dos times	<p><b>Usuário</b> – Poderá consultar as estatísticas dos times através de requisições API via uma interface. A API retornará os dados da tabela "estatísticas_times" estruturados em formato JSON.</p> <p>Exemplo de resposta: {data:[{ id: 1, roubos: 8, wl: "10-2", bloqueios: 3, lancesLivres: 12, rebotes: 40, pontos: 85, assistencias: 20, cestasDentroPerimetro: 25, cestasTresPontos: 8 }]} </p>

# **ESTRUTURA - DOC DE VISÃO**

**9. Licenciamento e Instalação**

**10. Interligação com Outros Sistemas**

**11. Proposta de Solução Tecnológica Escolhida**

**12. Valor da Proposta**

**13. Cronograma de Execução**

**14. Termo de Aceitação**

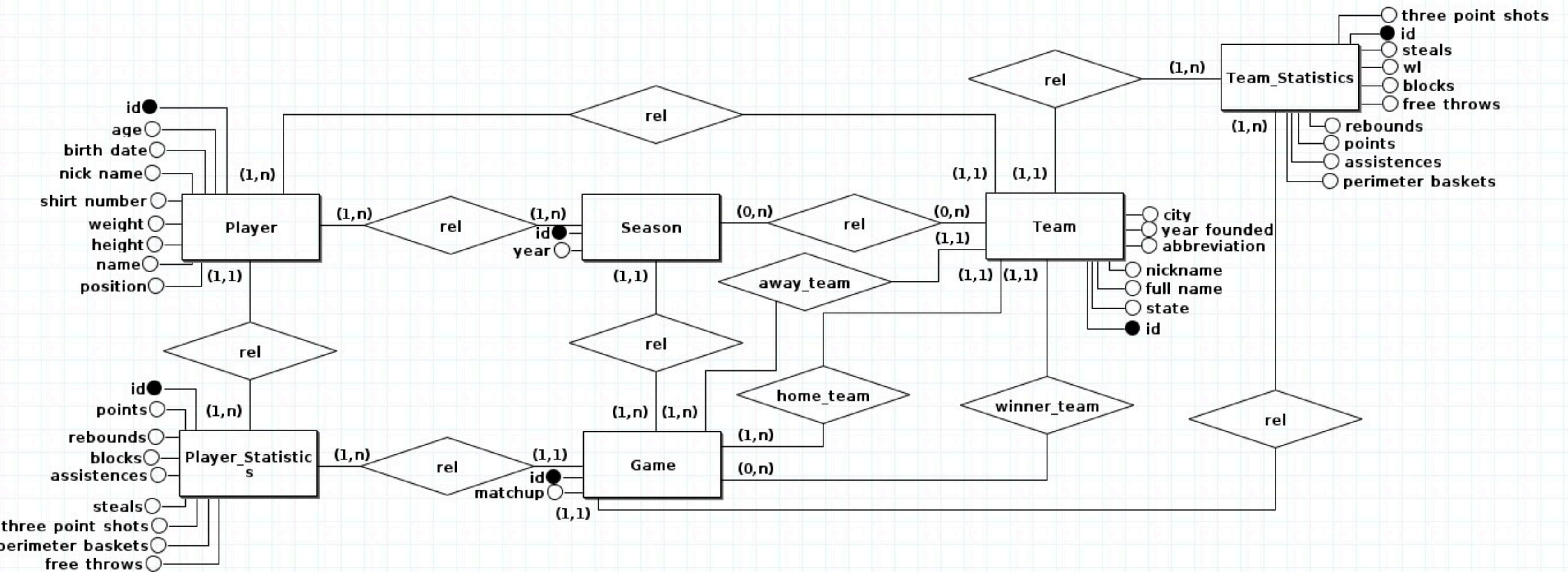
**15. Restrições**

**16. Documentação**

**17. Versionamento**

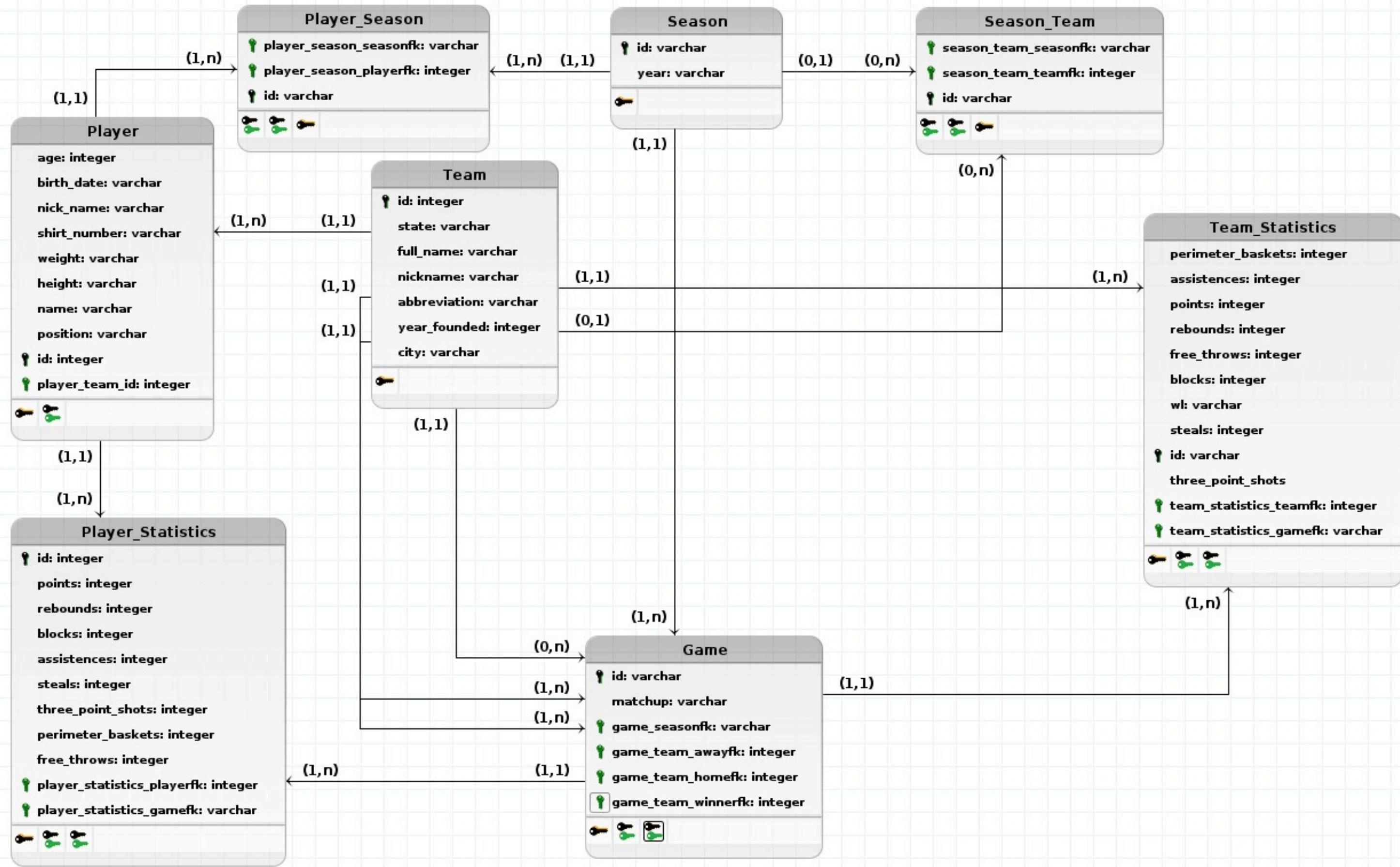
# ESQUEMA CONCEITUAL





# ESQUEMA LÓGICO





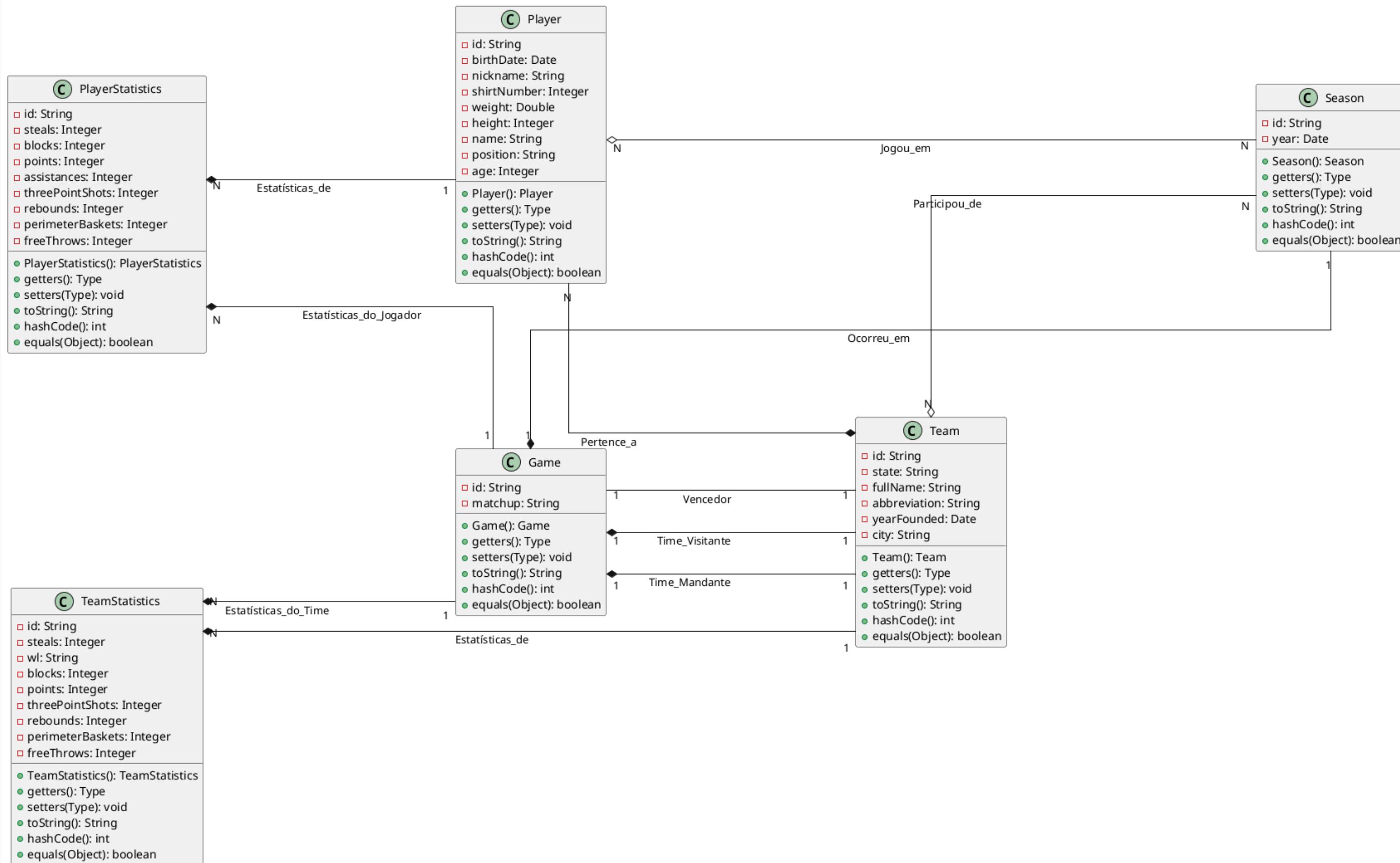
# DIAGRAMAS

LAKERS

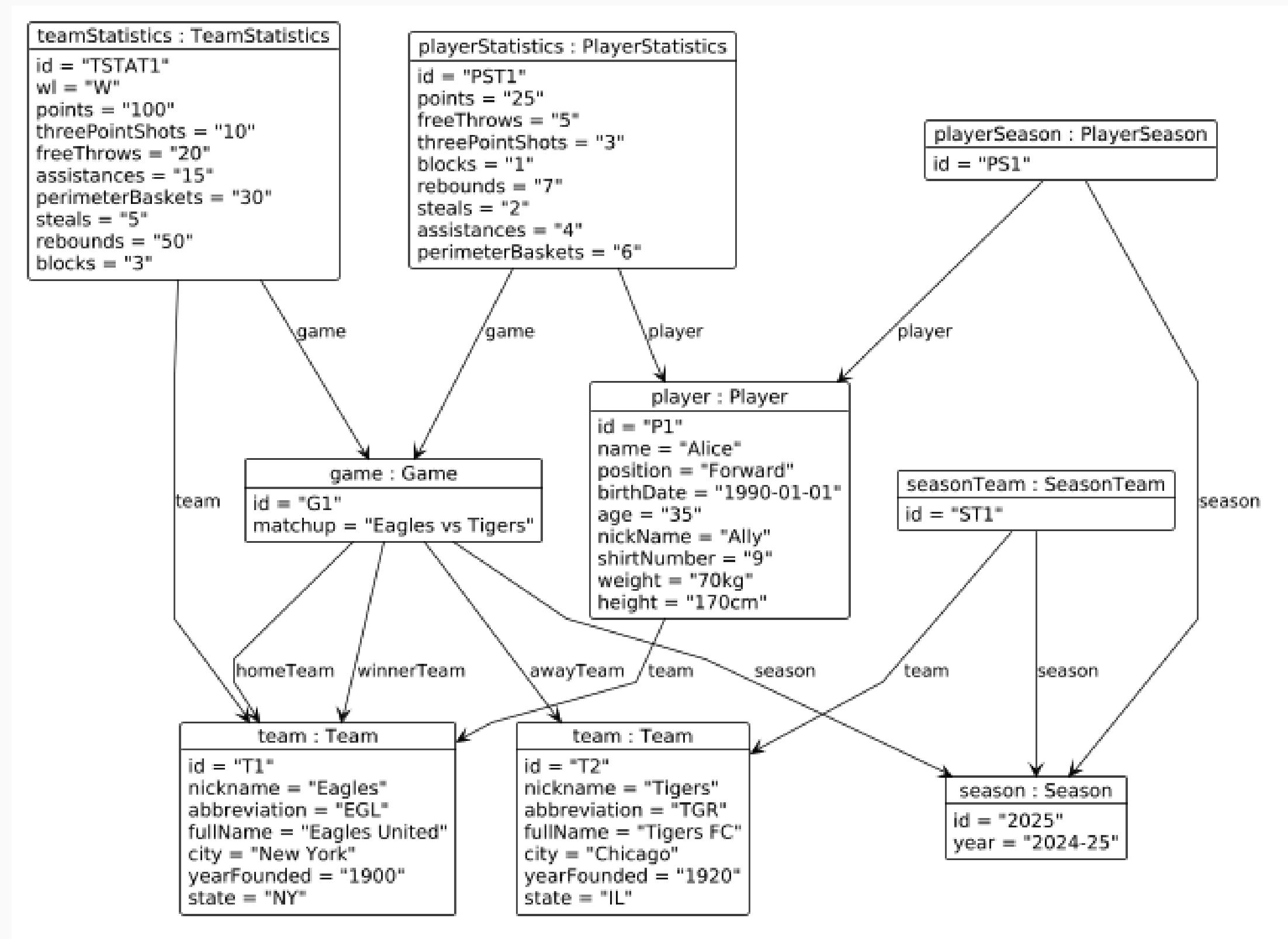
77



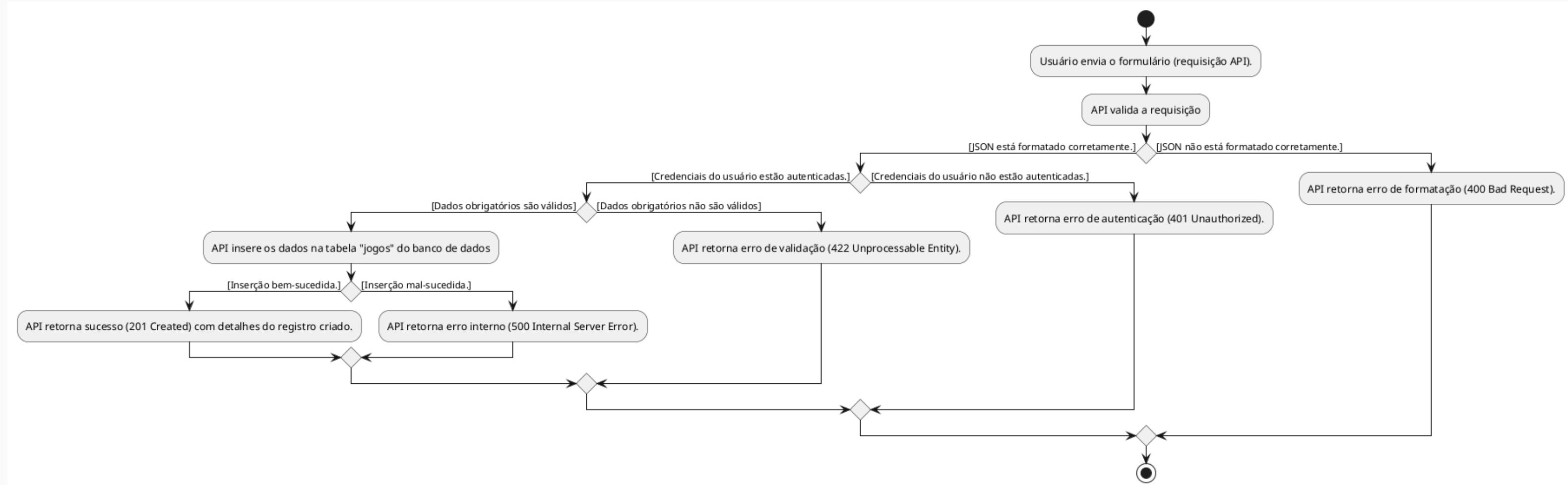
# DIAGRAMA DE CLASSE



# DIAGRAMA DE OBJETO

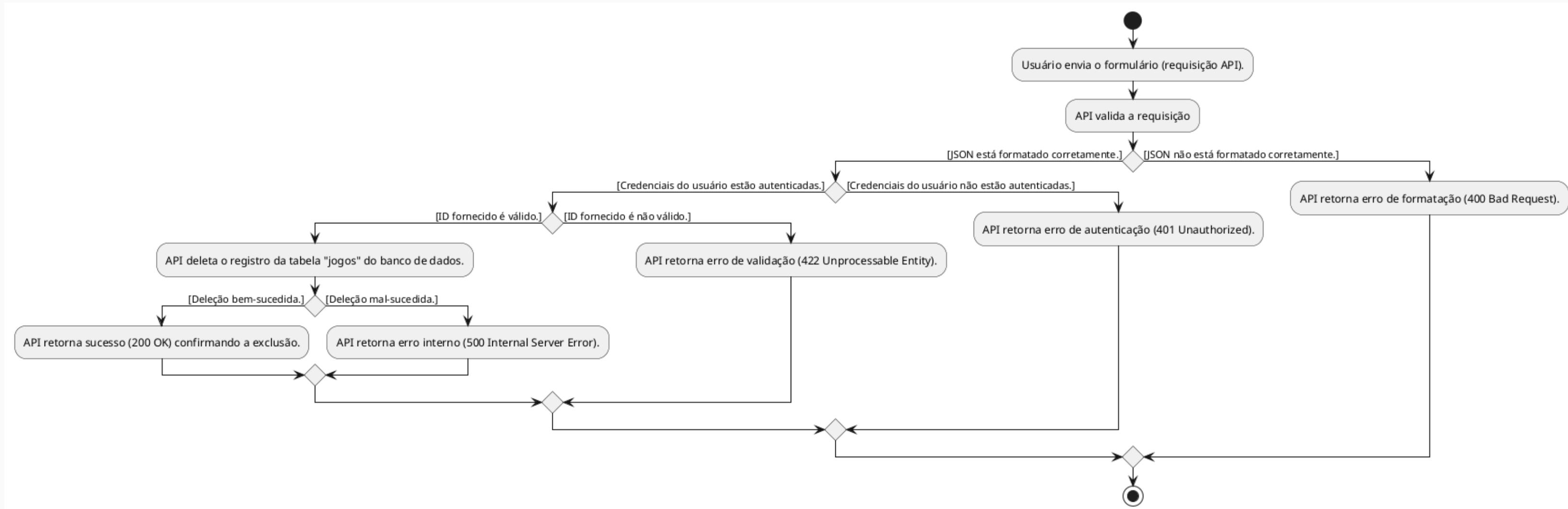


# DIAGRAMA DE ATIVIDADES



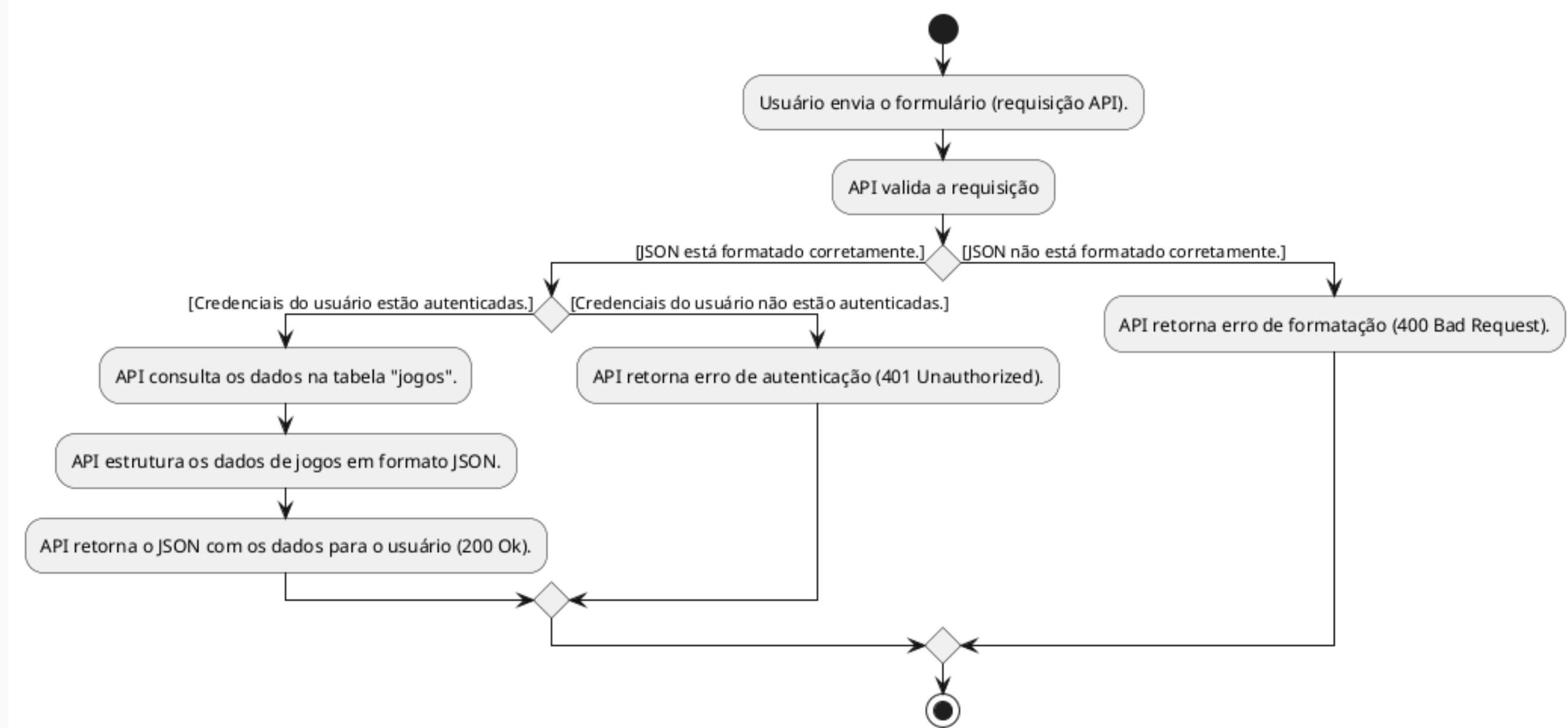
1- Diagrama de atividades referente a criação de um jogo

# DIAGRAMA DE ATIVIDADES



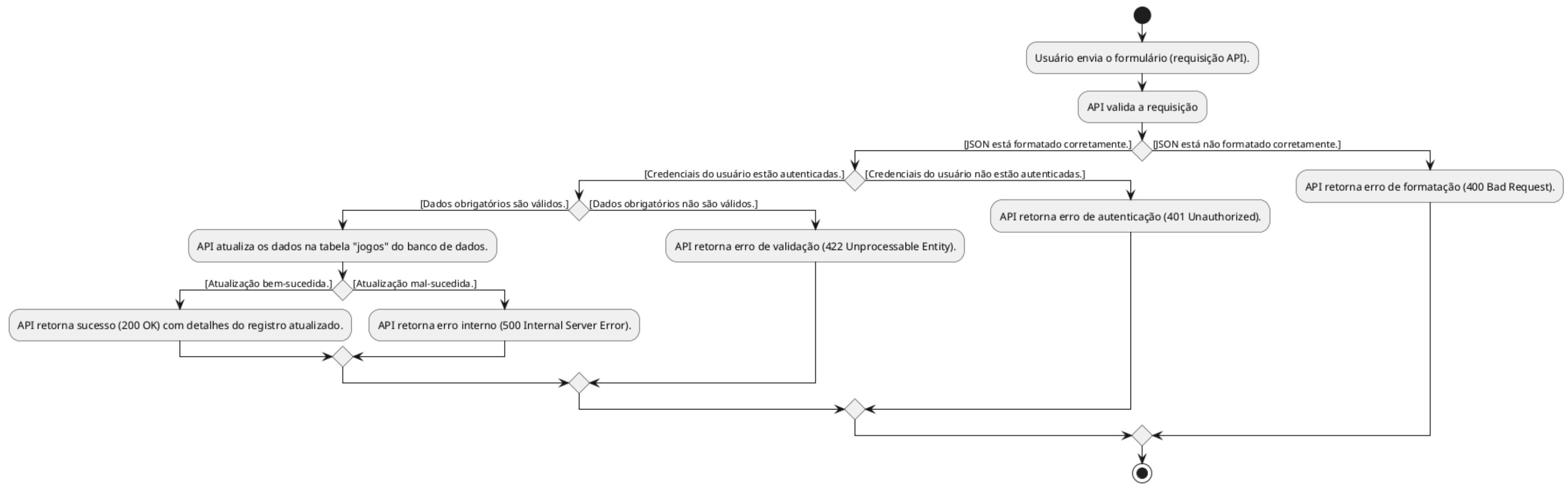
2- Diagrama de atividades referente a exclusão de um jogo

# DIAGRAMA DE ATIVIDADES



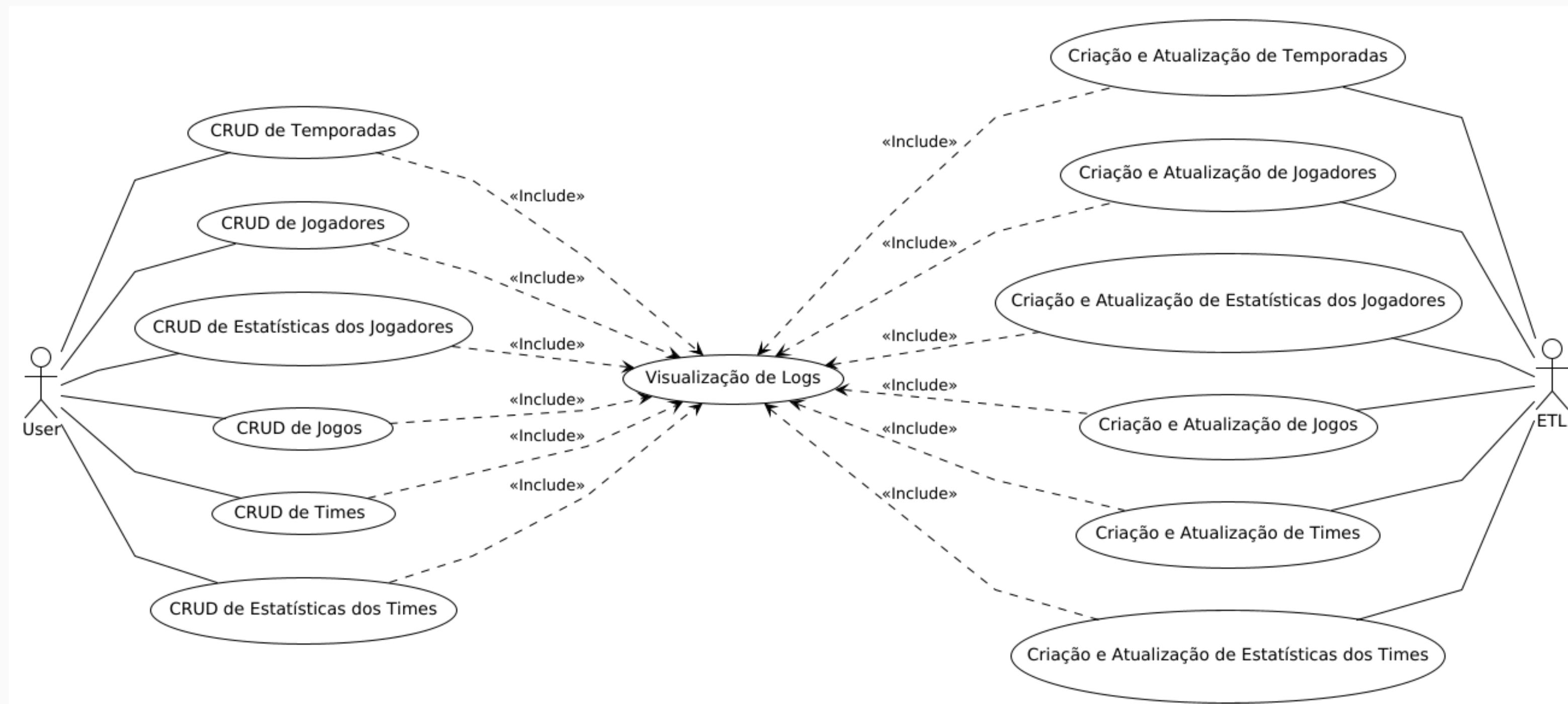
3- Diagrama de atividades referente a leitura de um jogo

# DIAGRAMA DE ATIVIDADES

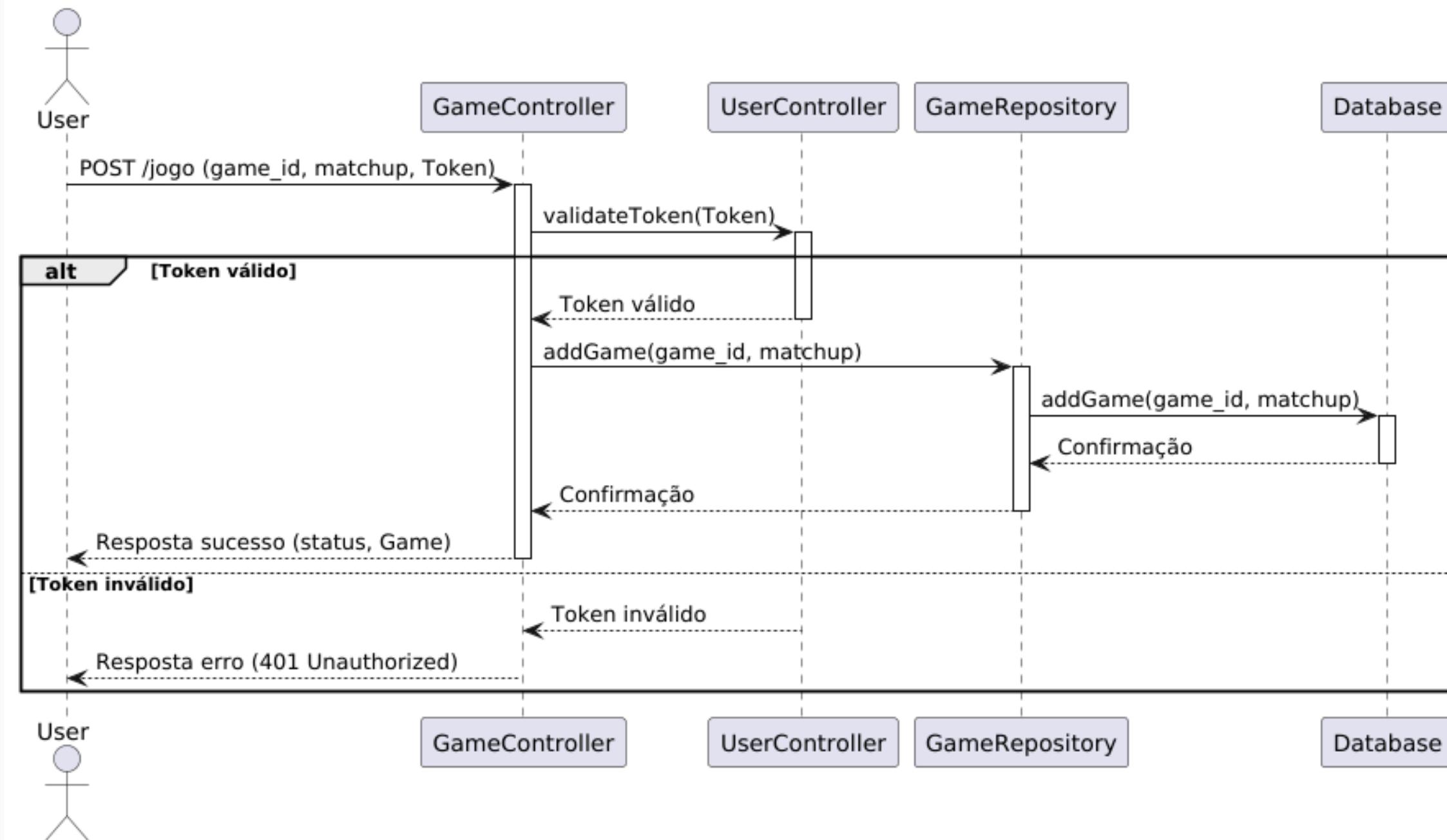


4- Diagrama de atividades referente a atualização de um jogo

# DIAGRAMA DE CASO DE USO

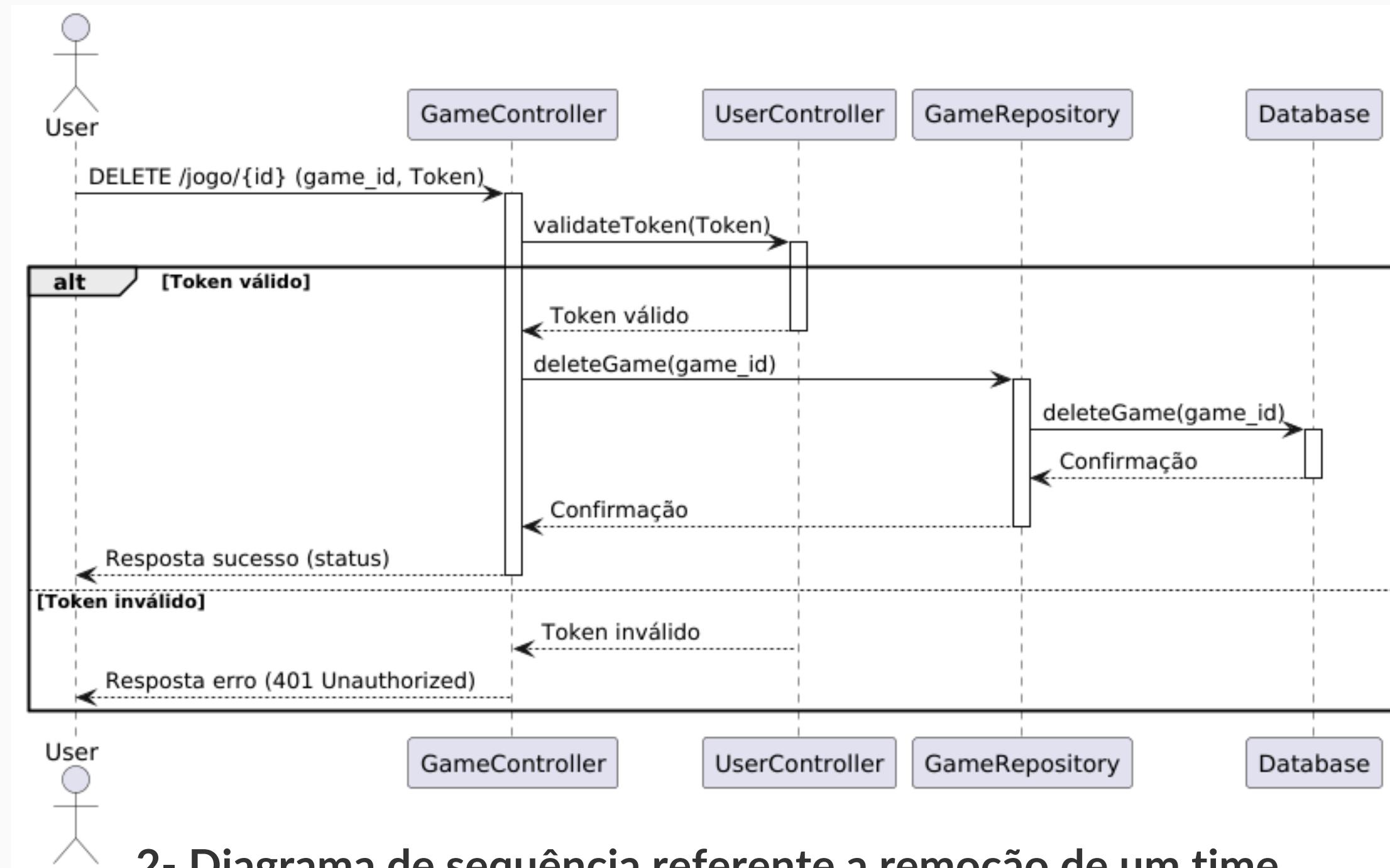


# DIAGRAMA DE SEQUÊNCIA



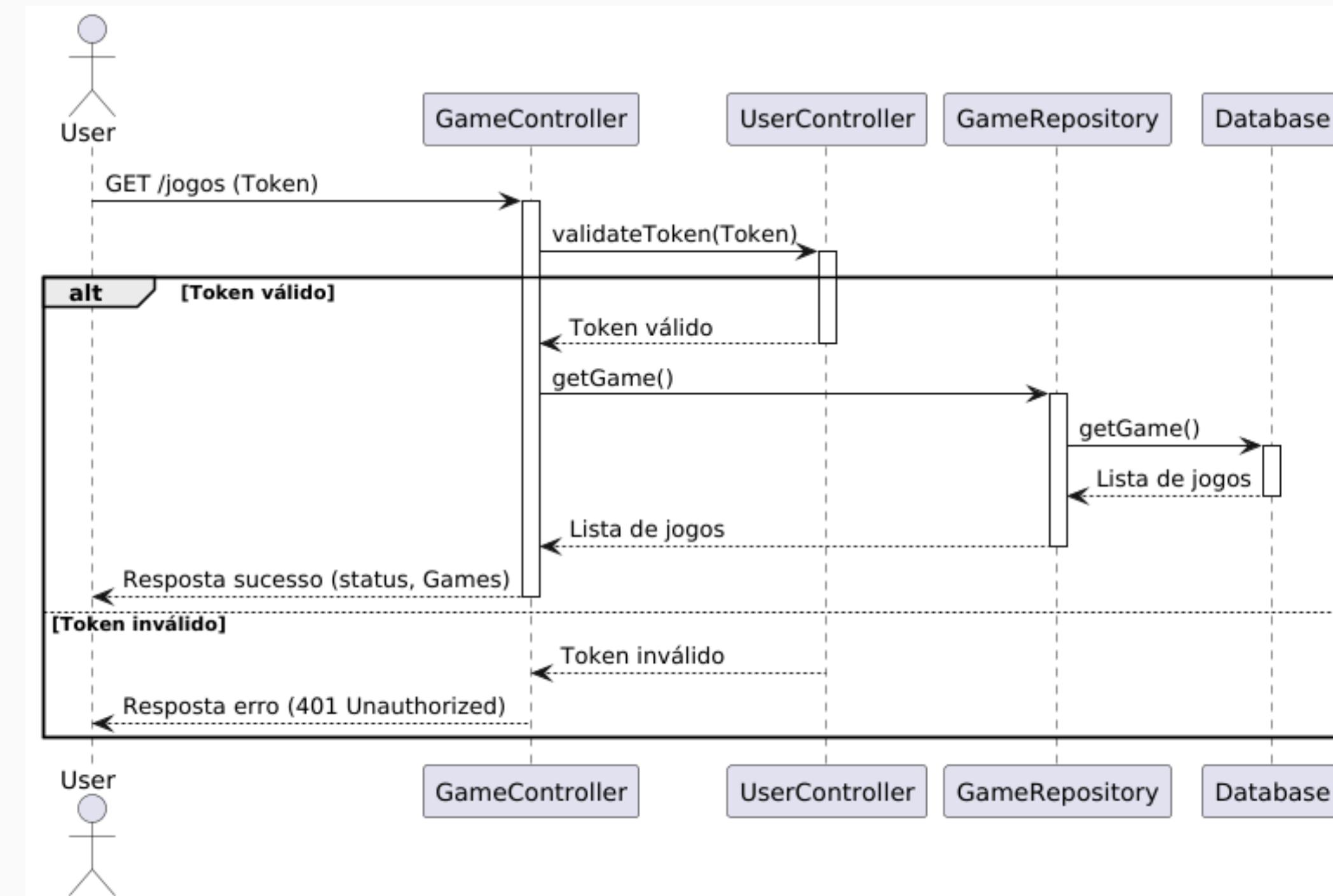
1- Diagrama de sequência referente a criação de um time

# DIAGRAMA DE SEQUÊNCIA



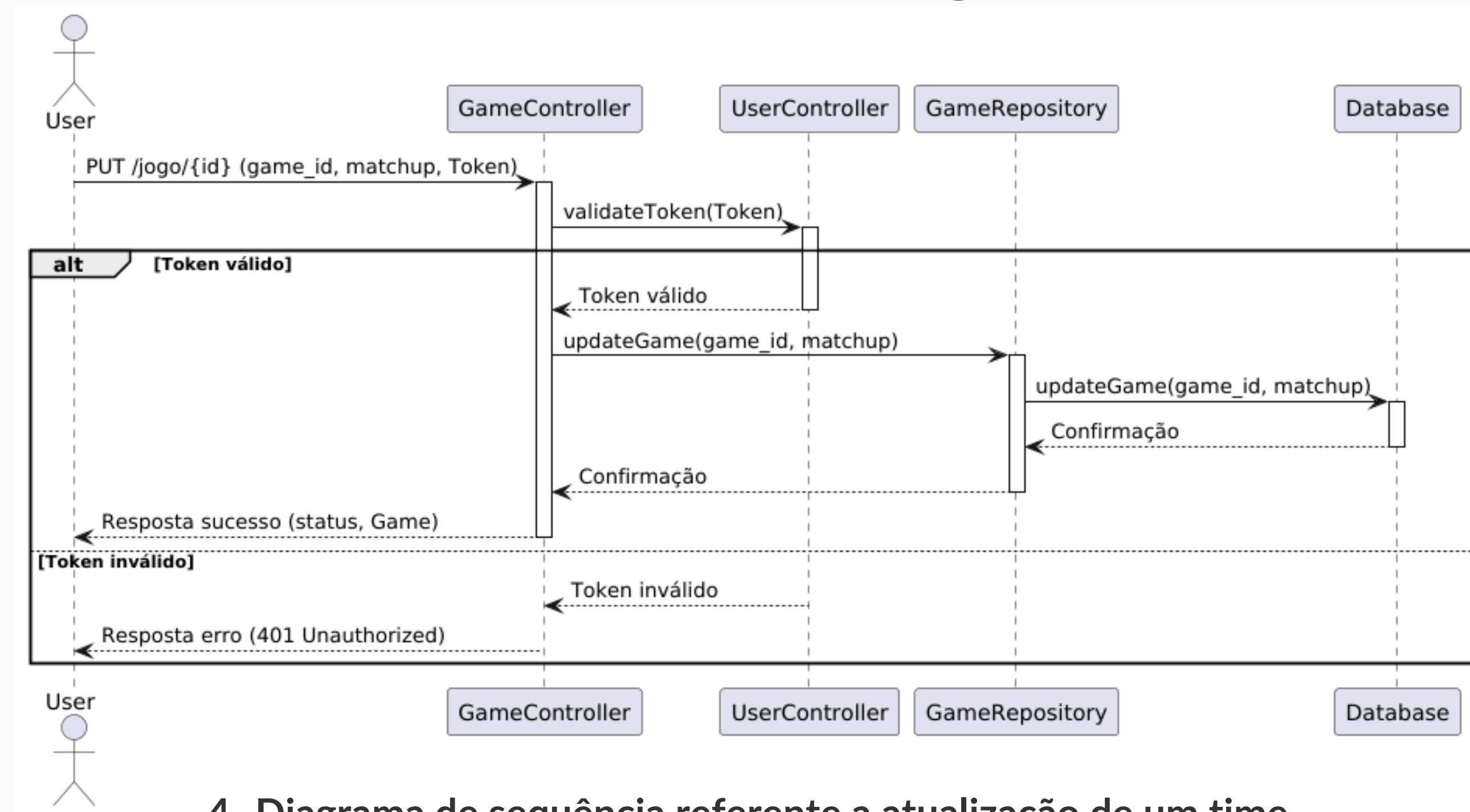
2- Diagrama de sequência referente a remoção de um time

# DIAGRAMA DE SEQUÊNCIA



3- Diagrama de sequência referente a leitura de um time

# DIAGRAMA DE SEQUÊNCIA



4- Diagrama de sequência referente a atualização de um time

# CONSISTÊNCIA DE DIAGRAMAS



# CRITÉRIOS UTILIZADOS

- 1 Classe vazia;
- 2 Diagrama vazio;
- 3 Mais de uma classe com o mesmo identificador;
- 4 Classe declarada abstrata, mas sem ter método abstrato;
- 5 Sobrecarga de métodos;
- 6 Classe declarada concreta, mas com método abstrato;
- 7 Falta de definição de tipo de atributo, parâmetro ou retorno de método;
- 8 Ciclo em relacionamentos de herança;

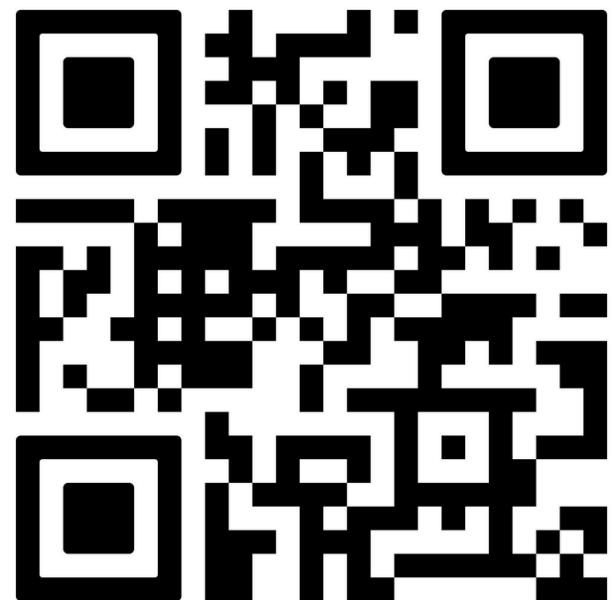
# CRITÉRIOS UTILIZADOS

- 9 Mais de um atributo com o mesmo identificador na mesma classe ou na mesma hierarquia de herança;
- 10 Presença de relacionamento de realização ligando uma classe a uma interface sem que a classe apresente todos os métodos declarados na interface;
- 11 Métodos abstratos herdados e não sobrescritos no nível mais baixo da hierarquia de herança de uma aplicação;

# LINKS ÚTEIS:



GITHUB



NOTION



# OBRIGADO!

