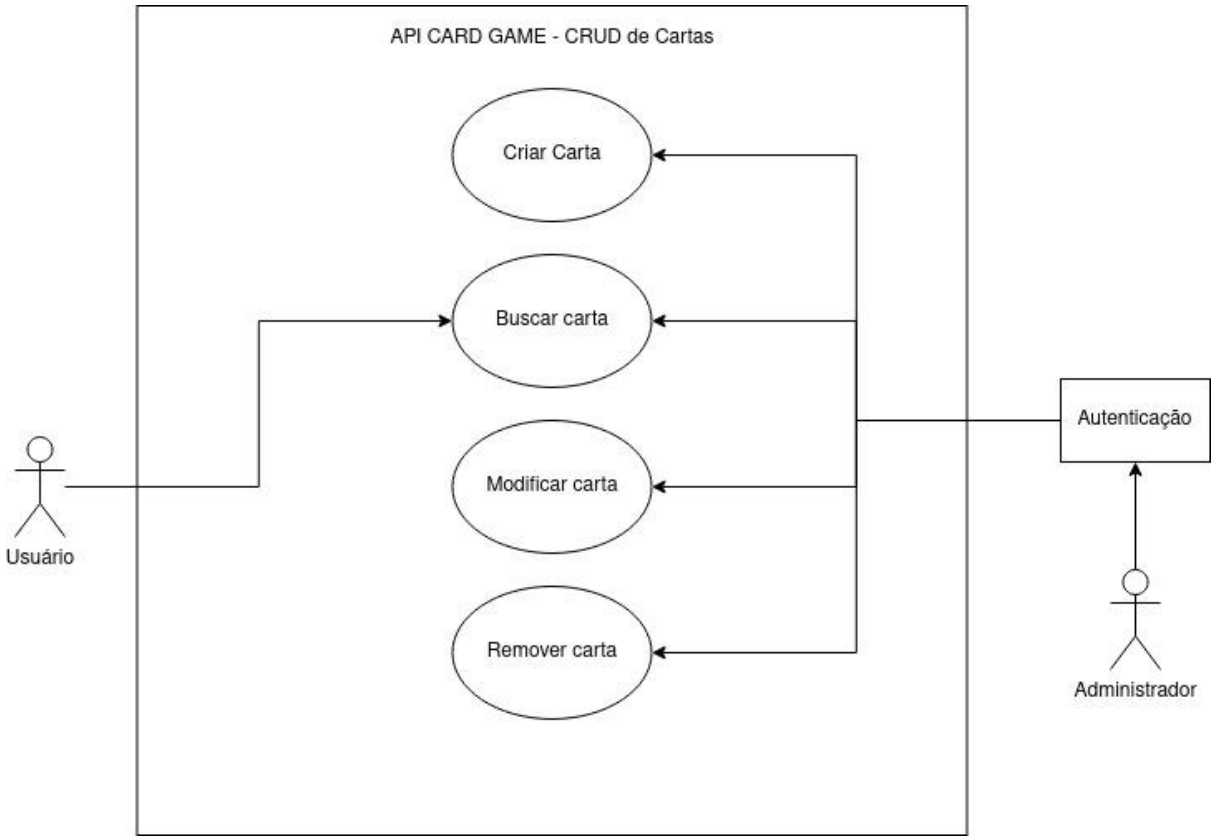
# **Projeto**: Card Game - API REST

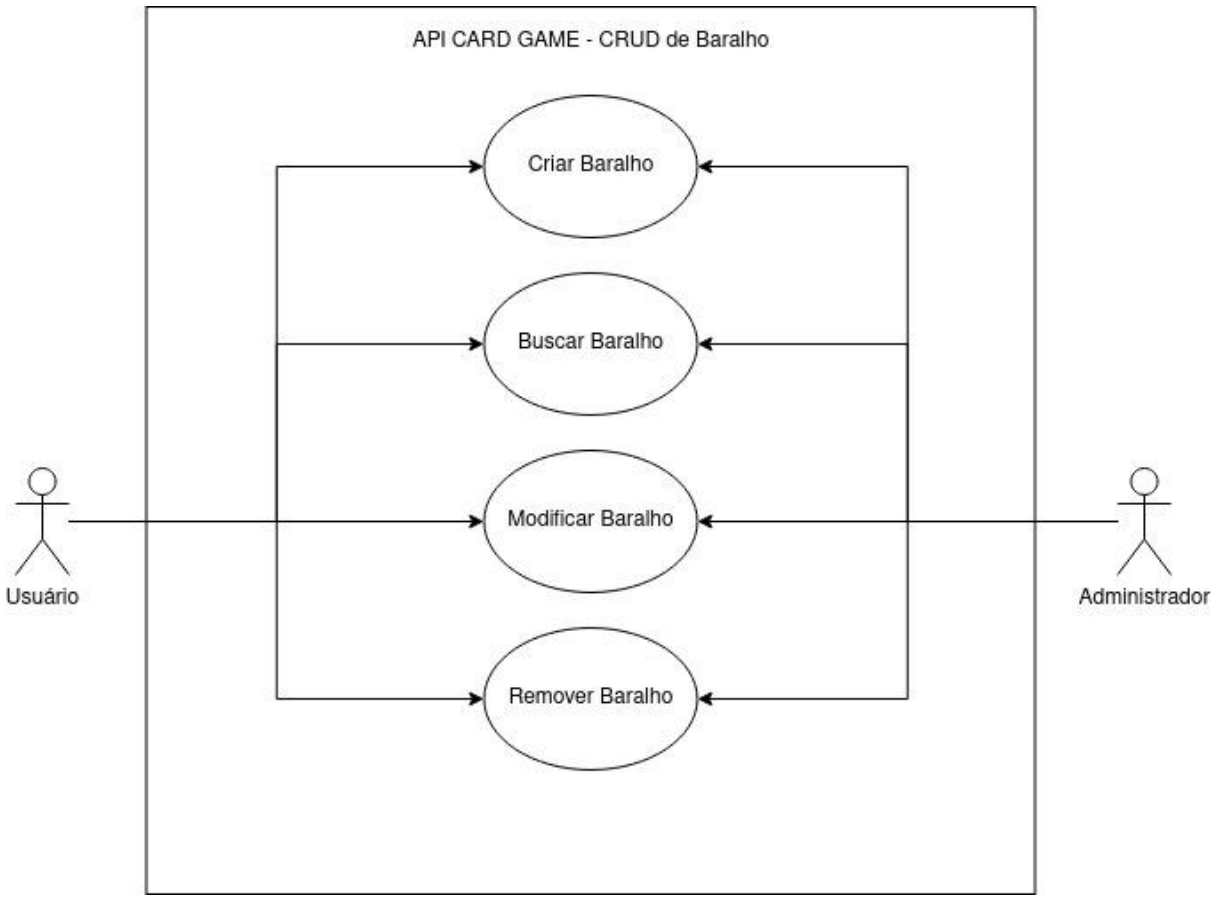
**Nome**: Luis Fernando Nunes

Para o desenvolvimento deste projeto será utilizado o modelo “Cascata”. A escolha desta metodologia se dá em conta que atualmente eu utilizo muito metodologias ágeis e por conta disso não me aprofundei muito no cascatão. Fora também que para um projeto pessoal onde não tem envolvimento com um cliente final e o objetivo é criar um produto bem definido e liberado para modificações externas, além de ser pequeno, se encaixa bem para utilizar e explorar este método.

Seguindo o modelo os requisitos já estão definidos em casos de uso (diagrama) e seguem a seguir:

**Requisitos do projeto:**






**Projeto do sistema:**

Quanto ao projeto do sistema, o hardware necessário para rodar com objetivo de criar projetos para tal é bem simples e pouco robusto. Tudo que será necessário é um servidor para rodar o projeto em Groovy, um DNS público para poder realizar as requisições e obter as respostas do servidor (Cadastro, leitura, modificação e remoção pela API) e acesso a um banco de dados em PostgreSQL. Ficará no projeto o SQL do banco utilizado na construção do projeto, porém o mesmo pode ser modificado para se encaixar melhor em futuros projetos. Pode ser também rodado de forma local para desenvolvimento, utilizando localhost.

**Projeto do software:**

O projeto busca a construção de uma API REST para card games (com um modelo de card game YU-GI-OH) onde poderá ser cadastrado, lidas, modificadas, removidas cartas e baralhos. A ideia é criar um backend livre para que outros desenvolvedores possam alimentar esta API em projetos semelhantes, podendo modificar de forma fácil para se adaptar ao que necessitarem. Para isso será focado o uso de código claro, limpo, organizado e com testes para garantir o funcionamento quando for (se for) modificado.

**Implementação:**

As etapas de implementação estão dividas em:

- Iniciar o projeto usando Spring boot.
- Realizar configurações necessárias para utilização do banco de dados PostgreSQL.
- Construir os endpoints (rotas) que serão chamadas
- Construir os serviços que serão responsáveis pelos tratamentos necessários serão chamados pelas rotas.
- Construir os repositórios que se comunicarão com o banco de dados.
- Criar testes unitários para cada parte desenvolvida.
- Revisar todos os tratamentos de exceções construídos em todo código.
- Criar testes de integração para garantir o funcionamento do projeto ao todo.
- Liberação de um readme para o GITHUB com a documentação da API e algumas informações importantes.

**Testes Unitários:**

Como consta na parte de implementações, serão criados testes unitários para garantir o funcionamento de cada parte do projeto (controller, service e repository). Assim garantimos o funcionamento a longo prazo das funcionalidades tanto para o projeto como está, tanto para modificações que possam acidentalmente quebrar alguma funcionalidade existente de forma indevida, servindo de garantia, facilitando aos desenvolvedores que modificaram a identificarem onde ocorreu o problema.

**Testes de Integração:**

Será criado para o final do projeto, teste de integração para garantir a funcionalidade do projeto como um todo.

**Operação e Manutenção:**

No fim, ficará o projeto completo disponível no GITHUB para utilização livre, podendo ser modificado para se adaptar aos projetos pessoais ou comerciais de cada um.