# VISÃO GERAL DO PROJETO

O projeto tem como objetivo desenvolver uma aplicação web do Jogo da Velha, na qual os usuários poderão criar perfis e poderão jogar uns contra os outros. A aplicação contará com um sistema ranqueado, no qual contará com uma classificação dos jogadores de acordo com o número de vitórias e a taxa de vitória.

Ao acessar a aplicação, o usuário poderá criar uma conta e buscar por uma partida contra pessoas que também estão acessadas ao mesmo tempo no servidor. Ao final de cada partida, as estatísticas dos jogadores serão atualizadas em tempo real.

A principal motivação deste projeto é aprimorar minhas habilidades de programação, utilizando como base os conhecimentos adquiridos na faculdade e em cursos extracurriculares, assim aprender novos conceitos. Como este é o meu primeiro grande projeto em desenvolvimento individualmente, é possível que o sistema passe por ajustes no planejamento, levando em consideração minhas capacidades e limitações.

# LEVANTAMENTO DOS REQUISITOS

Como ideia inicial do projeto e para planejar o desenvolvimento, foi realizado o levantamento dos requisitos funcionais e não funcionais do sistema que serão descritos a seguir. Os requisitos poderão sofrer alterações futuramente, podendo ser adicionados ou removidos.

## Requisitos funcionais

* **Registro de usuários:** Permitir que os usuários se cadastrem no site/aplicativo com nome de usuário, senha e informações de perfil;
* **Autenticação de usuários**: Permitir que os usuários façam login no sistema para acessar as funcionalidades do jogo;
* **Jogo da Velha**: Implementar a lógica do jogo da velha, incluindo um tabuleiro interativo onde os jogadores possam fazer suas jogadas;
* **Sistema de Ranqueada:** Registrar as estatísticas dos jogadores, como número de vitórias, derrotas, empates e taxa de vitória. Atualizar essas estatísticas após cada partida concluída;
* **Convite para partida:** Permitir que os jogadores convidem outros jogadores específicos para uma partida;
* **Busca de partida automática:** Implementar um sistema de *matchmaking* que encontre automaticamente um oponente para o jogador, com base no nível de habilidade ou outros critérios;
* **Histórico de partidas:** Registrar um histórico de partidas jogadas, incluindo resultados, jogadores envolvidos e datas.

## Requisitos não funcionais

* **Interface de usuário intuitiva:** O *front-end* deve ser amigável, fácil de usar e responsivo em diferentes dispositivos;
* **Portabilidade:** A aplicação deverá ser responsiva em diferentes dispositivos;
* **Segurança:** Garantir a segurança dos dados dos usuários e proteger contra ataques;
* **Desempenho:** O sistema deve ter um desempenho adequado, permitindo que os jogadores realizem suas jogadas de forma rápida e responsiva;
* **Escalabilidade:** O sistema deve ser capaz de lidar com um número crescente de usuários e partidas sem comprometer o desempenho;
* **Conectividade com o banco de dados:** Estabelecer a conexão com o banco de dados para armazenar informações do jogo e do perfil do usuário.

# DIAGRAMA DE CASO DE USO

Um caso de uso é uma técnica de modelagem utilizada para descrever a interação entre os atores (usuários ou sistemas externos) e o sistema que está sendo desenvolvido. Ele descreve como o sistema é usado para alcançar um objetivo específico do usuário, mostrando as principais funcionalidades e interações do sistema em termos de cenários de uso. No contexto deste projeto, o caso de uso é representado conforme ilustrado na Figura 1.

Figura 1 - Diagrama de caso de uso

Diagrama

Descrição gerada automaticamente

Fonte: Autoria própria (2023)

## Descrição dos casos de uso

Para compreender cada caso do uso, será apresentado a seguir a descrição detalhada de cada um dos casos de uso, conforme apresentados na Figura 1.

### Caso de uso Buscar partida

Tabela 1 - Explicação do caso de uso "Buscar partida"

|  |  |
| --- | --- |
| **Nome do caso de uso** | **Buscar partida** |
| Ator Principal | Jogador |
| Atores secundários |  |
| Resumo | Este caso de uso descreve as ações que ocorrem quando o jogador inicia a busca por uma partida de jogo da velha no sistema |
| Pré-condições | Nenhuma. |
| Pós-condições | Início da partida contra outro jogador |
| **Fluxo Principal** | |
| Ações do Ator | Ações do Sistemas |
| 1 Seleciona a opção “Buscar nova partida” |  |
|  | 2 O sistema procura por outras pessoas que estejam conectadas e buscando partidas na plataforma. |
|  | 3 O sistema encontra um novo jogador e uma nova partida é criada. |
|  | 4 O sistema apresenta o tabuleiro com todos os campos vazios para ambos os jogadores. |
| **Fluxo Alternativo I  - Cancelar busca por partida** | |
| Ações do Ator | Ações do Sistemas |
| 1 O jogador seleciona a opção “Cancelar busca” |  |
|  | 2 O sistema interrompe a procura pela partida e retorna o jogador para a tela inicial. |

Fonte: Autoria própria (2023)

### Caso de uso Realizar jogada

Tabela 2 - Caso de uso realizar jogada

|  |  |
| --- | --- |
| **Nome do caso de uso** | **Realizar jogada** |
| Ator Principal | Jogador |
| Atores secundários |  |
| Resumo | Este caso de uso descreve as ações que ocorrem quando o jogador inicia a busca por uma partida de jogo da velha no sistema |
| Pré-condições | Jogo iniciado |
| Pós-condições | Jogada realizada |
| **Fluxo Principal** | |
| Ações do Ator | Ações do Sistemas |
| 1 O jogador seleciona uma posição do tabuleiro. |  |
|  | 2 O sistema verifica se o campo selecionado está livre. |
|  | 3 Caso a posição esteja livre, se o jogador por o Jogador 1, a posição será preenchida com o X. Caso seja o Jogador 2, será preenchido com o O. |
|  | 4 O sistema verifica se o jogo terminou com vitória ou empate. |
|  | 5 Caso o jogo não tenha terminado, o sistema alterna a vez do jogador. |
| **Fluxo Alternativo I  - Jogada inválida** | |
| Ações do Ator | Ações do Sistemas |
| 1 O jogador seleciona um campo já preenchido no tabuleiro |  |
|  | 2 O sistema impede a jogada e solicita que o jogador da vez selecione outra posição, retornando para o passo 1 do fluxo principal. |
| **Fluxo Alternativo II  - Fim de jogo** | |
| Ações do Ator | Ações do Sistemas |
| 1 O jogador seleciona um campo no tabuleiro. |  |
|  | 2 O sistema realiza os passos 2 e 3 do fluxo principal. Caso uma das condições de vitória sejam atendidas ou não existirem mais posições livres, o sistema informa o final do jogo. |
|  | 3 O sistema registra o resultado da partida no histórico de ambos os jogadores. |
|  | 4 Ambos os jogadores são retornados para a tela inicial. |

Fonte: Autoria própria (2023).

# MODELAGEM E DESENVOLVIMENTO DO SISTEMA

# PROJETO DE INTERFACES

# PROJETO DO BANCO DE DADOS

# ARQUITETURA DA APLICAÇÃO

Para esta aplicação, serão utilizados dois tipos arquiteturas, uma para o *front-end* e outra para o *back-end***.**

## Arquitetura Back-end

No *back-end*, a arquitetura escolhida foi a arquitetura REST, que ajudará na separação de responsabilidades entre a interface e o armazenamento de dados, além de facilitar a comunicação entre serviços, sem comprometer o desempenho da aplicação. A Figura 2 demonstra uma visão geral de como essa arquitetura será implementada neste projeto.

Figura 2 - Arquitetura REST do sistema

Diagrama

Descrição gerada automaticamente

Fonte: Autoria própria (2023

Portanto, assim que a interface do sistema realizar uma solicitação, como uma jogada feita pelo usuário, por exemplo, será feito o seguinte fluxo de processamento no *back-end*:

- Os *controllers* serão responsáveis por receber as requisições e encaminhá-las para os *services*;

- Os *services*, por sua vez, vão receber a jogada, validar e atualizar o estado do tabuleiro de acordo com as regras do jogo da velha. Após o processamento, o *controller* envia a resposta à interface, exibindo o novo estado do tabuleiro caso a jogada tenha sido realizada com sucesso, ou informando e houve algum vencedor ou se a partida terminou em empate;

- Ao final de uma partida, existirão métodos dentro dos *services* que se comunicarão com as classes dos *repositories*. Com isso, os *repositories* irão acessar e manipular os dados dos jogadores envolvidos para atualizar as respectivas estatísticas de acordo com o resultado.

Assim, esse fluxo de processamento garante que as regras do jogo da velha sejam seguidas, que o estado do tabuleiro seja atualizado corretamente e que o jogador tenha uma experiência interativa e responsiva ao fazer suas jogadas, enquanto os *repositories* ajudam a armazenar e gerenciar os dados dos jogadores, possibilitando a atualização das estatísticas de acordo com o desfecho das partidas.

## Arquitetura Front-end

Para a interface, a arquitetura utilizada foi a de desenvolvimento baseado em componentes. Os componentes são um dos conceitos centrais do React, permitindo a reutilização de funcionalidades em várias partes do sistema. Além disso, essa abordagem oferece escalabilidade devido à modularidade dos componentes, tornando a manutenção da aplicação mais simples e promovendo o desacoplamento entre suas diferentes partes. Essa separação reduz o risco de efeitos colaterais em caso de falhas.

# PLANOS DE TESTE

# REFERÊNCIAS

FRANCO, Felipe. Adicionando autenticação do Google ao seu web app. **Fábrica de código**, 2022. Disponível em: <https://www.fabricadecodigo.com/adicionando-autenticacao-do-google-ao-seu-web-app/>. Acesso em: 12 de jul. de 2023.

MANOEL. JUnit Tutorial. **DevMedia**, 2009. Disponível em: <https://www.devmedia.com.br/junit-tutorial/1432>. Acesso em: 11 de jul. de 2023.

ARQUITETURA REST: Saiba o que é e seus diferenciais. **Totvs**, 2020. Disponível em: <https://www.totvs.com/blog/developers/rest/>. Acesso em: 19 de jul. de 2023.

BUILDING REST services with Spring. **Spring.io**, 2023. Disponível em: < https://spring.io/guides/tutorials/rest/>. Acesso em: 19 de jul. de 2023.

LIMA, Guilherme. REST: Conceito e fundamentos. **Alura**, 2020. Disponível em: <https://www.alura.com.br/artigos/rest-conceito-e-fundamentos>. Acesso em: 19 de jul. de 2023.

REST API Documentation. **IBM**, 2023. Disponível em: <https://www.ibm.com/docs/en/intelligent-promising?topic=apis-rest-api-documentation>. Acesso em: 19 de jul. de 2023.

REST e Java. **DevMedia,** 2023. Disponível em: <https://www.devmedia.com.br/guia/rest-e-java/36819>. Acesso em: 19 de jul. de 2023.

BARRO, Bruna. O que é DOM e quais suas vantagens. **Hostinger Tutoriais**, 2023. Disponível em: <https://www.hostinger.com.br/tutoriais/dom-o-que-e>. Acesso em: 20 de jul. de 2023.

NARDINI, Bruno. Como criar componentes React com uma arquitetura escalável usando Atomic Design. **Medium**, 2021. Disponível em: < https://medium.com/rd-shipit/como-criar-componentes-react-com-uma-arquitetura-escal%C3%A1vel-usando-atomic-design-74a67aaf47e0>. Acesso em: 20 de jul. de 2023.

YOUR first component. **React.dev**, 2023. Disponível em: < https://react.dev/learn/your-first-component>. Acesso em: 20 de jul. de 2023.