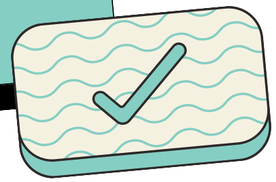
SMART GAMEPLAYS

FUNCIONALIDADES



01 CREATE (CRIAR)

02 READ (LISTAR)

03 UPDATE (EDITAR)

04 DELETE (APAGAR)

05 RANKING

06 PARTIDAS

OBJETIVOS

Este projeto é um sistema de gerenciamento de jogadores que permite criar, listar, editar, remover jogadores, registrar partidas, e exibir o ranking dos jogadores com base em seus pontos. O projeto é implementado em Node.js e utiliza a biblioteca `readline` para interação via linha de comando.


ANÁLISE
CRUD
C
R
U
D

ESTRUTURA

JOGADOR

Array que armazena os jogadores com suas informações (nome, jogo, pontos, vitórias, derrotas).

PARTIDAS

Array que registra as partidas realizadas entre os jogadores,aumentando vitórias e derrotas e diminuindo pontos no caso de derrotas.

RANKING

Array que será usado para exibir o ranking dos jogadores com base nos pontos.

C (CRIAR)& R (LISTAR)

Criar Jogador :Permite cadastrar um novo jogador com nome, jogo e pontos iniciais.

Listar Jogadores: Exibe a lista de jogadores cadastrados.

```
function cadastrarJogador() {
    rl.question('Digite o nome do jogador: ', (nome) => {
        rl.question('Digite o jogo do jogador: ', (jogo) => {
            rl.question('Digite os pontos do jogador: ', (pontos) =>{
                jogador.push({nome: nome, jogo: jogo, pontos: parseInt(pontos), vitoria: 0 , derrota: 0})
                console.log('Jogador cadastrado com sucesso!')
                exibirMenu()
            })
        })
    })
}
function listarJogadores() {
    if(jogador == '') {
        console.log('Não há jogadores cadastrados')
            exibirMenu()
        }else{
            console.log('Há jogadores cadastrados')
            for(let i = 0; i < jogador.length; i++){
                console.log(jogador[i])
        }exibirMenu()
    }
}
```

U (ATUALIZAR)

Editar Jogador: Permite editar as informações de um jogador existente.

```
function editarJogador(){
    if(jogador.length==0) {
        console.log('Não há jogadores para editar.')
    }else{
        console.log('Lista de jogadores')
        jogador.forEach((jogador, index) => {
          console.log(`${index + 1}. ${jogador.nome}`)
        })
        rl.question('Digite o número do jogador: ', (numero) => {
            if(numero > 0 && numero <= jogador.length){
                rl.question('Digite o novo jogador: ', (nome) => {
                    rl.question('Digite o  jogo: ', (jogo) =>{
                        rl.question('Digite os pontos do novo jogador: ', (pontos) =>{
                            jogador[numero - 1] = {
                                nome,
                                jogo,
                                pontos
                            }
                            exibirMenu()
                        })
                    })
                })
            }else{
                console.log('Numero não reconhecido,retornando...')
                exibirMenu()
            }
        })
    }
}
```

D (DELETAR)

Remover Jogador: Remove um jogador da lista de jogadores.

```
function removerJogador() {
    if (jogador.length == 0) {
        console.log('Não há jogadores para remover.')
        exibirMenu()
    } else {
            console.log('Lista de Jogadores:')
            jogador.forEach((jogador, index) => {
                console.log(`${index + 1}. ${jogador.nome}`)
            })
            rl.question('Digite o jogador que deseja remover: ', (remover) => {
                if(remover > 0 && remover<= jogador.length) {
                    jogador.splice (remover -1, 1)
                    console.log('Jogador removido com sucesso!')
                    exibirMenu()
                }else {
                    console.log('Opção Inválida, digite novamente.')
                    exibirMenu()
                }
        })
    }
}
```

PARTIDAS

Registrar Partida: Registra uma partida entre dois jogadores, atualizando os pontos, vitórias e derrotas.

```
function partidaJogador() {
    if (jogador.length === 0) {
        console.log('Não há jogadores cadastrados!');
        exibirMenu();
    } else {
        console.log('Lista de jogadores');
        jogador.forEach((jogador, index) => {
            console.log(`${index + 1}. ${jogador.nome}`);
        });

        rl.question('Qual o jogador venceu a partida? (número): ', (jog1) => {
            const vencedorIndex = parseInt(jog1) - 1;
            if (vencedorIndex >= 0 && vencedorIndex < jogador.length) {
                rl.question('Qual o jogador perdeu a partida? (número): ', (jog2) => {
                    const perdedorIndex = parseInt(jog2) - 1;
                    if (perdedorIndex >= 0 && perdedorIndex < jogador.length) {
                        jogador[vencedorIndex].pontos++;
                        jogador[vencedorIndex].vitoria++;
                        jogador[perdedorIndex].pontos--;
                        jogador[perdedorIndex].derrota++;
                        console.log('Partida realizada!');
                    } else {
                        console.log("Número inválido para perdedor, retornando...");
                    }
                    exibirMenu();
                });
            } else {
                console.log("Número inválido para vencedor, retornando...");
                exibirMenu();
            }
        });
    }
}
```

RANKING

Ranking dos Jogadores: Exibe o ranking dos jogadores com base em seus pontos.

```
function rankingJogador() {
    if (jogador.length === 0) {
        console.log('Não há jogadores cadastrados!');
        exibirMenu();
    } else {
        let ranking = jogador.slice().sort((a, b) => b.pontos - a.pontos);
        ranking = ranking.slice(0, 5);
        console.log("O ranking é:");
        ranking.forEach((jog, index) => {
            console.log(`${index + 1}º: ${jog.nome} - ${jog.pontos} pontos`);
        });

        exibirMenu();
    }
}
```

EXTRA

Lions Gameplays


OBRIGADO