

# **PROPOSTA DE APLICATIVO E DIAGRAMA DE CLASSES**

**TÍTULO:** UNB Sinuqueiros.

**GRUPO:** 7.

**INTEGRANTES:** Gustavo Mello Tonnera (211055272), Vinícius da Silva Araújo (221001981).

## **1. Descrição do problema**

Atualmente na Universidade de Brasília (UNB), diversos Centros Acadêmicos (CA) possuem mesas de sinuca, algumas melhores do que outras. Com certa frequência, a fim de adquirir verba para outras atividades, os CAs promovem torneios de sinuca. Entretanto, como os administradores dos CAs também são estudantes da UNB, a organização dessas competições é um grande desafio, haja vista a enorme quantidade de tempo que as disciplinas demandam dos alunos. Em virtude desse problema, propomos o aplicativo UNB Sinuqueiros para ajudar os administradores dos CAs a organizarem tais competições.

## **2. Regras de Negócio**

O aplicativo UNB Sinuqueiros possui como principal objetivo facilitar a criação, organização e inscrição de torneios de sinuca na UNB. Dessa maneira, cada usuário precisa criar um cadastro para utilizar o aplicativo e poder se inscrever ou criar competições. Para realizar o cadastro, o usuário deve fornecer as seguintes informações: nome completo, curso, email, senha e tipo de usuário (organizador ou participante). Ao efetuar o login no aplicativo, cada usuário pode visualizar as competições que estão sendo organizadas e suas informações, podendo se inscrever em nenhuma ou várias delas, caso seja do tipo participante, ou podendo criar/organizar torneios, caso seja do tipo organizador. Além disso, cada usuário pode criar suas próprias competições.

Cada competição possui os seguintes atributos: nome, descrição, local, data, horário, número máximo de inscritos, conjunto de regras, número de partidas por jogo, um organizador e uma lista de participantes. É responsabilidade do organizador atualizar as informações do torneio, como os resultados dos jogos. Ademais, uma competição pode ter dois formatos: mata-mata ou liga. Na competição mata-mata, será feito um chaveamento aleatório entre os participantes e cada jogo eliminará um participante. Na competição do tipo liga, existe um sistema de pontos definido pelo organizador do torneio e todos os participantes se enfrentam pelo menos 1 vez.

### 3. Diagrama de Classes

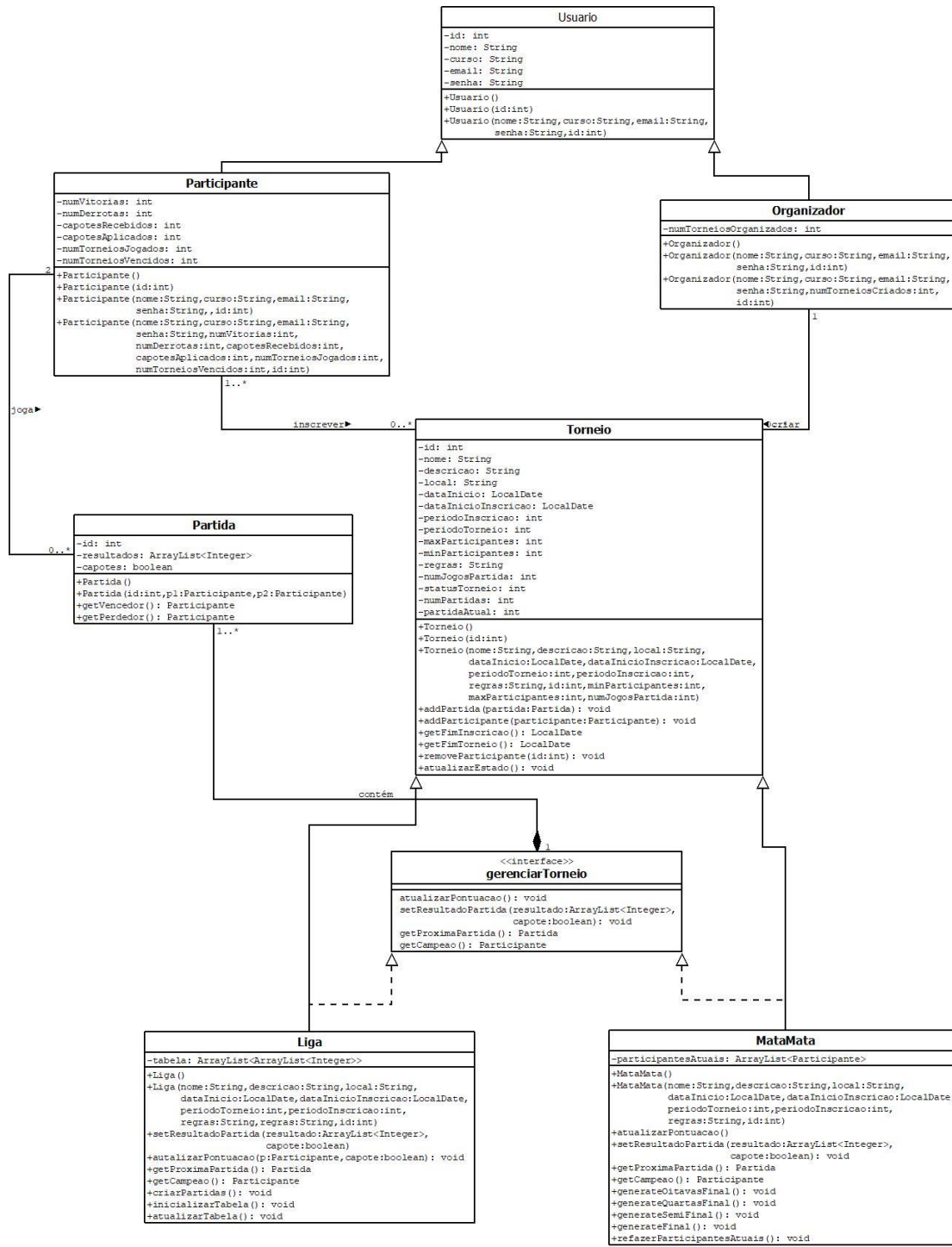


Figura 1 – Versão final do Diagrama de Classes

## **4. Descrição das Classes**

### **4.1 Classes presentes no diagrama**

A classe “Usuario” é uma classe abstrata pai das classes “Participante” e “Organizador”, a qual representa um usuário do aplicativo. Essa classe possui os seguintes atributos: “id” (int)- representa o id do usuário-, “nome” (String)- representa o nome do usuário-, “curso” (String)- representa o id do usuário-, “email” (String)- representa o email cadastrado do usuário- e “senha” (String)- representa a senha cadastrada do usuário. Cada um desses atributos possui seus respectivos métodos getter e setter. Além desses métodos, a classe sobrescreve o método toString e implementa três construtores: o construtor padrão, um construtor que inicializa apenas o valor do atributo id e um construtor que inicializa o valor de todas os atributos da classe.

A classe “Participante” é uma das classes filhas da classe “Usuario”. Ela representa o tipo de usuário jogador, o qual participa e se inscreve nos torneios e disputa partidas. Essa classe possui os seguintes atributos: “numVitorias” (int)- representa o número de partidas vencidas por esse jogador-, “numDerrotas” (int)- representa o número de partidas perdidas por esse jogador-, “capotesRecebidos” (int)- representa o número de capotes recebidos por esse jogador-, “capotesAplicados” (int)- representa o número de capotes aplicados por esse jogador-, “numTorneiosJogados” (int)- representa o número de torneios jogados por esse jogador- e “numTorneiosVencidos” (int)- representa o número de torneios vencidos por esse jogador, além dos atributos da classe “Usuario” descritos anteriormente. Cada um desses atributos possui seus respectivos métodos getter e setter. Ademais, essa classe implementa quatro construtores: o construtor padrão, um construtor que inicializa apenas o valor do atributo id, um construtor que inicializa apenas os valores das variáveis da classe pai e um construtor que inicializa o valor de todas os atributos.

A classe “Organizador” é a outra classe filha da classe “Usuario”. Ela representa o usuário do tipo organizador, cujas funções é criar e administrar torneios. Essa classe possui um único atributo- além dos atributos da classe pai-, o atributo “numTorneiosCriados” (int) que representa o número de torneios que um organizador já criou. Esse atributo possui seu respectivo método getter e setter. Além do mais, essa classe implementa três construtores: o construtor padrão, um construtor que inicializa apenas os atributos da classe pai e um que inicializa todas os atributos.

A classe “Torneio” é a classe pai de outras duas classes, cuja função é modelar um torneio de sinuca. Essa classe possui os seguintes atributos: “id” (int)- representa o id do torneio-, “minParticipantes” (int)- representa o número mínimo de participantes do torneio-, “maxParticipantes” (int)- representa o número máximo de participantes do torneio-, “statusTorneio” (int)- representa o estado do torneio-, “numJogosPartida” (int)- representa o número de jogos por partida do torneio-, “periodoInscricao” (int)- representa o período de inscrição do torneio em dias-, “periodoTorneio” (int)- representa o período em que as partidas do torneio serão disputadas em dias-, “nome” (String)- representa o

nome do torneio-, “descricao” (String)- representa a descrição do torneio-, “local” (String)- representa o local no qual as partidas do torneio serão disputadas-, “regras” (String)- representa o conjunto de regras usados nas partidas do torneio-, “dataInicio” (LocalDate)- representa a data de início do torneio-, “dataInicioInscricao” (LocalDate)- representa a data de início das inscrições-, “participantes” (ArrayList<Participante>)- representa uma lista dos participantes inscritos no torneio-, “organizador” (Organizador)- representa o organizador do torneio-, “partidas” (ArrayList<Partida>)- representa uma lista das partidas do torneio-, “numPartidas” (int)- representa o número de partidas que serão disputadas- e “partidaAtual” (int)- representa o índice da partida a ser disputada na lista de partidas. Cada um desses atributos possui seus respectivos métodos getter e setter. Além disso, essa classe implementa três construtores: o construtor padrão, um construtor que inicializa apenas o atributo “id” e um construtor que inicializa todos os atributos da classe, e outros seis métodos: “addPartida”- método responsável por adicionar uma nova partida no atributo “partidas”-, “addParticipante”- método responsável por adicionar um novo participante no atributo “participantes”-, “removeParticipante”- método responsável por remover um participante do atributo “participantes”-, “getFimInscricao”- método que retorna a data do fim das inscrições do torneio-, “getFimTorneio”- método que retorna a data do fim do torneio- e “atualizarStatus”- método responsável por atualizar o estado do torneio levando em consideração a data do dia de hoje.

A interface “gerenciarTorneio” foi implementada para padronizar os métodos das classes filhas da classe “Torneio”. Essa interface possui três métodos: “getProximaPartida”, “getCampeao” e “setResultadoPartida”, cujas funções são, respectivamente, retornar a próxima partida a ser disputada, retornar o campeão do torneio e atribuir um resultado a partida atual do torneio em questão.

A classe “Liga” é uma das classes filha da classe “Torneio”, implementa a interface “gerenciarTorneio” e possui o objetivo de modelar um torneio do tipo liga, como o “Brasileirão” – Campeonato Brasileiro de Futebol. Nesse tipo de torneio, cada participante joga uma vez contra os outros jogadores e cada vitória garante 3 pontos ao vencedor. Assim, o participante que tiver o maior número de pontos ao final de todas as partidas vence o campeonato, sendo os critérios de desempate o número de vitórias e, em seguida, o número de capotes aplicados. O número mínimo de participantes desse tipo de torneio é 10 e o máximo é 20. Essa classe possui um único atributo, além dos atributos da classe pai, denominado “tabela” (ArrayList<ArrayList<Integer>>), o qual representa a tabela de pontuação do torneio. Esse atributo possui seus respectivos métodos getter e setter. Ademais, essa classe implementa dois construtores: o construtor padrão e um construtor que inicializa todos os atributos da classe pai e da classe filha, implementa os três métodos da interface “gerenciarTorneio” e outros quatro métodos: “atualizarPontuacao”- método responsável por atualizar a tabela dado o resultado de uma partida-, “criarPartidas”- método responsável por criar todas as partidas do torneio-, “inicializarTabela”- método que inicializa as linhas da tabela de pontuação com os valores iguais a zero para todos os participantes- e “atualizarTabela”- método que atualiza a tabela de pontuação de acordo com as partidas que já aconteceram.

A classe “MataMata” é a outra classe filha da classe “Torneio”, também implementa a interface “gerenciarTorneio” e possui o objetivo de modelar um torneio do tipo mata-mata, como a Copa do Brasil. Nesse tipo de torneio, os participantes se enfrentam em partidas eliminatórias, cujos vencedores avançam para a próxima fase e os perdedores são desclassificados. Assim, o participante que vencer a final vence o campeonato. O número mínimo de participantes desse tipo de torneio é 9 e o máximo é 16. Essa classe também possui apenas um único atributo, além dos atributos da classe pai, denominado “participantesAtuais” (ArrayList<Participante>), o qual representa a lista de jogadores que passam de fase em cada etapa do torneio. Esse atributo possui seus respectivos métodos getter e setter. Ademais, essa classe implementa dois construtores: o construtor padrão e um construtor que inicializa todos os atributos da classe pai e da classe filha, implementa os três métodos da interface “gerenciarTorneio” e outros cinco métodos: “generateOitavasFinal”- método responsável por gerar as oitavas de final-, “generateQuartasFinal”- método responsável por gerar as quartas de final -, “generateSemiFinal”- método responsável por gerar as semifinais-, “generateFinal”- método responsável por gerar a final- e “refazerParticipantesAtuais”- método responsável por gerar o atributo “participantesAtuais” a partir das partidas que já aconteceram.

A última classe presente no diagrama é a classe “Partida”, a qual é a responsável por modelar uma partida de sinuca entre dois participantes em um torneio. Essa classe possui os seguintes atributos: “id” (int)- representa o id da partida-, “resultado” (ArrayList<Integer>)- representa o resultado da partida-, “capote” (boolean)- representa se o perdedor perdeu sem derrubar nenhuma bola, “p1” (Participante)- representa o jogador 1-, “p2” (Participante)- representa o jogador 2-, “vencedor” (Participante)- representa o vencedor da partida- e “perdedor” (Participante)- representa o perdedor da partida. Desconsiderando os atributos “perdedor” e “vencedor” que possuem apenas o método getter, os demais atributos possuem seus respectivos métodos getter e setter. Além disso. Essa classe implementa dois construtores: o construtor padrão e um construtor que inicializa todas as variáveis da classe.

#### **4.2 Classes não presentes no diagrama**

A classe “ComparadorLinha” foi criada para ordenar as linhas das tabelas dos objetos da classe “Liga”. Ela possui um único método denominado “compare” usado para ordenar as linhas usando o número de pontos, número de vitórias e o número de capotes aplicados, respectivamente, como critérios de ordenação.

A classe “ConnectDB” foi criada para estabelecer a conexão da aplicação com o banco de dados. Ela possui diversos métodos responsáveis por alterar, deletar e criar objetos das classes “Partida”, “Organizador”, “Participante”, “Liga” e “MataMata” no banco de dados. Para mais informações da classe, ler o arquivo “ConnectDB.java”, o qual possui todos os métodos da classe “ConnectDB” documentados no início do arquivo. As classes “OrganizadorService”, “ParticipanteService”, “PartidaService” e “TorneioService” utilizam métodos dessa classe para se comunicar com o banco de dados.

A classe “OrganizadorService” foi criada para realizar operações relacionadas a classe “Organizador” no banco de dados. Ela possui cinco métodos: “createOrganizador”, “findOneOrganizador”, “findAllOrganizador”, “deleteOrganizador” e “updateOrganizador”, responsáveis por armazenar uma nova instância da classe “Organizador” no banco de dados, encontrar um objeto da classe “Organizador” no banco de dados dado um determinado “id”, encontrar todos os objetos da classe “Organizador” armazenados no banco de dados, deletar um objeto “Organizador” armazenado no banco de dados dado um “id” e alterar um objeto “Organizador” presente no banco de dados, respectivamente.

A classe “ParticipanteService” foi criada para realizar operações relacionadas a classe “Participante” no banco de dados. Ela possui cinco métodos: “createParticipante”, “findOneParticipante”, “findAllParticipante”, “deleteParticipante” e “updateParticipante”, responsáveis por armazenar uma nova instância da classe “Participante” no banco de dados, encontrar um objeto da classe “Participante” no banco de dados dado um determinado “id”, encontrar todos os objetos da classe “Participante” armazenados no banco de dados, deletar um objeto “Participante” armazenado no banco de dados dado um “id” e alterar um objeto “Participante” presente no banco de dados, respectivamente.

A classe “PartidaService” foi criada para realizar operações relacionadas a classe “Partida” no banco de dados. Ela possui quatro métodos: “createPartida”, “findOnePartida”, “addPartidaResult” e “deletePartida”, responsáveis por armazenar uma nova instância da classe “Partida” no banco de dados, encontrar um objeto da classe “Partida” no banco de dados dado um determinado “id”, alterar o resultado de um objeto “Partida” armazenado no banco de dados e deletar um objeto “Partida” armazenado no banco de dados dado um “id”.

A classe “TorneioService” foi criada para realizar operações relacionadas a classe “Participante” no banco de dados. Ela possui sete métodos: “createTorneio”, “findOneTorneio”, “findAllTorneio”, “deleteTorneio”, “findAllTorneioParticipante”, “findAllTorneioPartida” e “updateTorneio”, responsáveis por armazenar uma nova instância da classe “MataMata” ou “Liga” no banco de dados, encontrar um objeto da classe “MataMata” ou “Liga” no banco de dados dado um determinado “id”, encontrar todos os objetos das classes “MataMata” e “Liga” armazenados no banco de dados, deletar um objeto “MataMata” ou “Liga” armazenado no banco de dados dado um “id”, encontrar todos os participantes de um torneio dado um “id”, achar todas as partidas de um torneio dado um “id” e alterar um objeto “MataMata” ou “Liga” presente no banco de dados, respectivamente.


A classe “TrabalhoFinal” é a classe principal do aplicativo e seu método main é usado para iniciar a aplicação. Além desse método, essa classe possui diversos outros métodos que se comunicam com o aplicativo, como “login” (realiza o login do usuário), “createUsuario” (cadastra um novo usuário) e “checkLogin” (verifica se o email e a senha

da tela de login estão cadastradas). Ela possui, também, atributos fundamentais para o funcionamento da aplicação. São eles: “login”- armazena o usuário que efetuou o login no aplicativo-, “currentTournamentId”- “id” do próximo torneio a ser criado-, “currentOrganizadorId”- “id” do próximo organizador a ser cadastrado-, “currentParticipanteId”- “id” do próximo participante a ser cadastrado-, “currentTorneio”- torneio cujas informações estão sendo visualizadas pelo usuário na tela de informações do torneio, “currentPartidaId”- “id” da próxima partida a ser criada, “torneios”- lista de torneio presentes no banco de dados, “participantes”- lista de participantes armazenados no banco de dados- e “organizadores”- lista de organizadores presentes no banco de dados. Para mais informações dos métodos da classe “TrabalhoFinal”, ler documentação presente no início do arquivo “TrabalhoFinal.java”.

## **5. Telas**

### **5.1 Tela de Login**

É a primeira tela que o usuário possui contato ao iniciar a aplicação. Nela, o usuário possui a opção de preencher os campos de email e senha para realizar o login no aplicativo ou clicar em “clique aqui” para ser direcionado para a tela de cadastro para criar uma conta no aplicativo. Caso o usuário digite um email e/ou uma senha que não está cadastrado, a mensagem “Usuário não encontrado!” é exibida. Caso o usuário não preencha todos os campos e tente fazer login, a mensagem “Por favor, preencha todos os campos!” é exibida.



A imagem mostra a tela de login do aplicativo "UnB Sinuqueiros". No topo, o nome do aplicativo "UnB Sinuqueiros" está em negrito. Abaixo, há um formulário com o título "Login". O formulário contém dois campos de entrada: "Email" com o exemplo "Ex: robertinho@gmail.com" e "Senha" com pontos para ocultar o texto. Abaixo dos campos, há um link "Novo usuário? clique aqui" e um botão "Login".

Figura 2 – Tela de Login

### **5.2 Tela de Cadastro**

Nessa tela o usuário possui a opção de criar um cadastro preenchendo os campos: nome, email, curso, email e tipo. O campo tipo define o tipo de conta a ser criada, ou de jogador ou de organizador. Contas do tipo jogador só podem participar dos torneios,

enquanto que contas do tipo organizador apenas organizam torneios. Caso o usuário insira um email fora do formato, a mensagem “Email inválido!” é exibida. Caso o usuário não preencha todos os campos e tente fazer login, a mensagem “Por favor, preencha todos os campos!” é exibida. Ademais, o usuário possui a opção de clicar em “clique aqui” para voltar a tela de login.

A imagem mostra a interface de cadastro do sistema "UnB Sinuqueiros". O formulário, intitulado "Cadastro", contém campos para "Nome" (com o exemplo "Ex: Robertinho"), "Email" (com o exemplo "robertinho@gmail.com"), "Curso" (com o exemplo "Ciência da Computação") e "Senha". Abaixo dos campos, há um menu suspenso com a opção "jogador" selecionada. No rodapé do formulário, há o link "Já tem conta? clique aqui" e um botão "Cadastrar".

Figura 3 – Tela de Cadastro

### **5.3 Tela de Início**

Assim que o usuário realiza o login no aplicativo, ele é direcionado para a tela de início, a qual consiste em uma imagem e alguns botões: botão “Lista de Torneios”, botão “Perfil” e botão “Criar Torneio”. O botão “Criar Torneio” só é visível para contas do tipo organizador. Cada botão ao ser pressionado, encaminha o usuário para uma nova tela: tela de lista de torneios (botão “Lista de Torneios”), tela de perfil (botão “Perfil”) ou tela de criação de torneios (botão “Criar Torneio”).

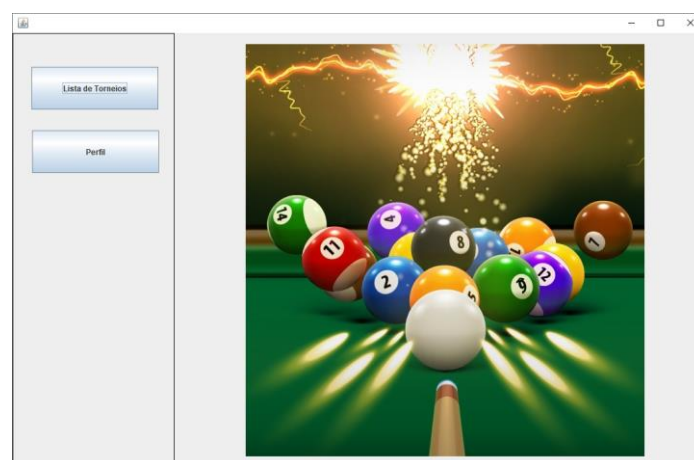


Figura 4 – Tela de Início para contas do tipo Jogador





Figura 5 - Tela de Início para contas do tipo Organizador

## 5.4 Tela de Lista de Torneios

Assim que o usuário clica no botão “Lista de Torneios” da tela de início, ele é redirecionado para a tela de lista de torneios, a qual consiste em uma tabela com todos os torneios presentes no banco de dados. Ao clicar em um torneio da tabela, uma nova tela é criada com as informações do torneio selecionado. É possível abrir apenas uma tela de informações do torneio por vez.

ID	Nome	Local	Início das Inscrições	Início
0	Mata-Mata Test 0	Cacomp	2023-03-29	2023-04-09
1	Mata-Mata Test 1	Cacomp	2023-01-21	2023-02-01
2	Mata-Mata Test 2	Cacomp	2023-01-29	2023-02-09
3	Mata-Mata Test 3	Cacomp	2023-04-05	2023-04-16
4	Mata-Mata Test 4	Cacomp	2023-05-12	2023-05-23
5	Mata-Mata Test 5	Cacomp	2023-03-26	2023-04-06
6	Mata-Mata Test 6	Cacomp	2023-05-31	2023-06-11
7	Mata-Mata Test 7	Cacomp	2023-06-03	2023-06-14
8	Mata-Mata Test 8	Cacomp	2023-05-16	2023-05-27
9	Mata-Mata Test 9	Cacomp	2023-03-05	2023-03-16
10	Liga Test 0	Cacomp	2023-02-24	2023-03-07
11	Liga Test 1	Cacomp	2023-04-09	2023-04-20
12	Liga Test 2	Cacomp	2023-05-10	2023-05-21
13	Liga Test 3	Cacomp	2023-03-03	2023-03-14
14	Liga Test 4	Cacomp	2023-02-05	2023-02-16
15	Liga Test 5	Cacomp	2023-02-21	2023-03-04
16	Liga Test 6	Cacomp	2023-05-17	2023-05-28
17	Liga Test 7	Cacomp	2023-01-18	2023-01-29
18	Liga Test 8	Cacomp	2023-03-06	2023-03-17
19	Liga Test 9	Cacomp	2023-01-01	2023-01-12

Figura 6 – Tela de Lista de Torneios

## 5.5 Tela de Perfil

Assim que o usuário clica no botão “Perfil” da tela de início, ele é redirecionado para a tela de perfil. A tela de perfil contém informações da conta do usuário (nome, curso, email, id e tipo de conta) e estatísticas de torneio (número de torneios criados, número de torneios jogados, número de torneios vencidos, número de vitórias, número de derrotas, número de capotes aplicados e número de capotes recebidos).

**Perfil**

**Estatísticas**  
Torneios criados: 0  
Torneios jogados: 1  
Torneios vencidos: 0  
Capotes recebidos: 0  
Capotes aplicados: 0  
Vitórias: 0  
Derrotas: 1

Id: 0  
Nome: Gustavo0  
Email: gustavo0@gmail.com  
Curso: Ciência da Computação  
Tipo: Jogador

Figura 7 – Tela de Perfil

## **5.6 Tela de Criação de Torneio**

Assim que o usuário clica no botão “Criar Torneio” da tela de início, ele é redirecionado para a tela de criação de torneio. Nessa tela, os organizadores de torneio podem criar torneios, especificando um nome, uma descrição, a regra dos jogos, o tipo de torneio, as datas do início das inscrições e do início dos jogos e os períodos de inscrição e de duração dos jogos. Caso o usuário não preencha todos os campos e tente fazer login, a mensagem “Por favor, preencha todos os campos!” é exibida. Caso o usuário preencha os campos com datas inválidas, a mensagem “Por favor, digite uma data válida!” é exibida.

**Criar Torneio**

Nome:   
Local:   
Regra:   
Início das Inscrições:   
Período de inscrições (dias):

Descrição:   
Tipo:   
Início dos jogos:   
Período dos jogos (dias):

Figura 8 – Tela de Criação de Torneios

## **5.7 Tela de Informações do Torneio**

Assim que o usuário clica em um torneio da tela de lista de torneios, uma nova tela aparece com as informações do torneio. Além disso, essa tela possui alguns botões: botão “Lista de Participantes”, botão “Pontuação”, botão “Inscrever-se”, botão “Cancelar Inscrição” e botão “Atualizar Resultados”. Os botões “Inscrever-se” e “Cancelar

Inscrição” são visíveis apenas para jogadores, enquanto que o botão “Atualizar Resultados” é visível apenas para organizadores. Os botões “Lista de Participantes”, “Pontuação” e “Atualizar Resultados”, ao serem pressionados, encaminham o usuário para uma nova tela: tela de lista de participantes, tela de pontuação do torneio e tela de atualização de resultados, respectivamente. Dependendo do estado do torneio, as inscrições ficam bloqueadas e alguns botões param de aparecer para qualquer usuário.

**Torneio do Tio Zé** Atualizar Resultados

Descrição

Local: jLabel1 Início das Inscrições: jLabel1

Regra: jLabel1 Encerramento das Inscrições: jLabel1

Tipo: jLabel1 Início dos Jogos: jLabel1

Organizador: jLabel13 Fim dos Jogos: jLabel1

Máximo de participantes: jLabel1 Número de jogos por partida: jLabel1

Mínimo de participantes: jLabel1

Lista de participantes Pontuação Inscrever-se Cancelar inscrição

Figura 9 – Tela de Informações do Torneio (imagem do NetBeans)

**5.8 Tela de Lista de Participantes**

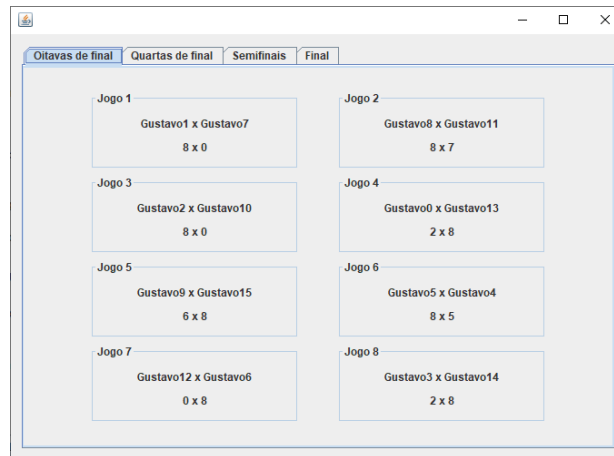
Assim que o usuário clica no botão “Criar Torneio” da tela de informações do torneio, uma nova tela aparece com a lista de participantes do torneio em questão.

ID	Nome	Curso
0	Gustavo0	Ciência da Computação
1	Gustavo 1	Ciência da Computação
2	Gustavo2	Ciência da Computação
3	Gustavo3	Ciência da Computação
4	Gustavo4	Ciência da Computação
5	Gustavo5	Ciência da Computação
6	Gustavo6	Ciência da Computação
7	Gustavo7	Ciência da Computação
8	Gustavo8	Ciência da Computação
9	Gustavo9	Ciência da Computação
10	Gustavo10	Ciência da Computação
11	Gustavo11	Ciência da Computação
12	Gustavo12	Ciência da Computação
13	Gustavo13	Ciência da Computação
14	Gustavo14	Ciência da Computação
15	Gustavo15	Ciência da Computação

Figura 10 – Tela de Lista de Participantes

**5.9 Tela de Pontuação do Torneio**

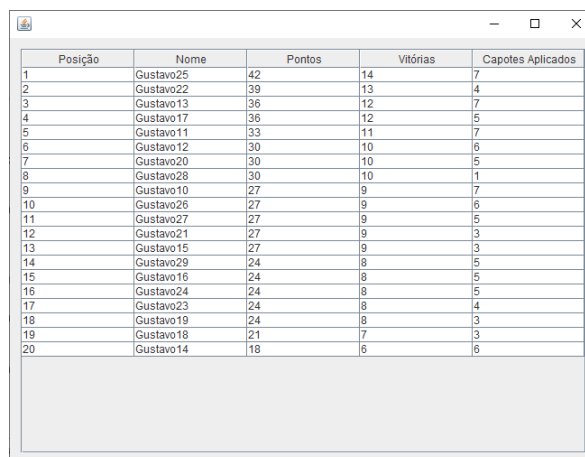
Assim que o usuário clica no botão “Pontuação” da tela de informações do torneio, uma nova tela aparece com a pontuação do respectivo torneio. Como existem dois tipos de torneio (mata-mata e liga), existem dois tipos de tela de pontuação.



The screenshot shows a window titled 'Pontuação (Mata-mata)' with a tabbed interface. The 'Oitavas de final' tab is selected. It displays eight games in a bracket format:

Jogo	Resultado
Jogo 1: Gustavo1 x Gustavo7	8 x 0
Jogo 2: Gustavo8 x Gustavo11	8 x 7
Jogo 3: Gustavo2 x Gustavo10	8 x 0
Jogo 4: Gustavo0 x Gustavo13	2 x 8
Jogo 5: Gustavo9 x Gustavo15	6 x 8
Jogo 6: Gustavo5 x Gustavo4	8 x 5
Jogo 7: Gustavo12 x Gustavo6	0 x 8
Jogo 8: Gustavo3 x Gustavo14	2 x 8

Figura 11 – Tela de Pontuação (Mata-mata)



The screenshot shows a window titled 'Pontuação (Liga)' displaying a league table with 20 rows. The table has five columns: Posição, Nome, Pontos, Vitórias, and Capotes Aplicados.

Posição	Nome	Pontos	Vitórias	Capotes Aplicados
1	Gustavo25	42	14	7
2	Gustavo22	39	13	4
3	Gustavo13	36	12	7
4	Gustavo17	36	12	5
5	Gustavo11	33	11	7
6	Gustavo12	30	10	6
7	Gustavo20	30	10	5
8	Gustavo28	30	10	1
9	Gustavo10	27	9	7
10	Gustavo26	27	9	6
11	Gustavo27	27	9	5
12	Gustavo21	27	9	3
13	Gustavo15	27	9	3
14	Gustavo29	24	8	5
15	Gustavo16	24	8	5
16	Gustavo24	24	8	5
17	Gustavo23	24	8	4
18	Gustavo19	24	8	3
19	Gustavo18	21	7	3
20	Gustavo14	18	6	6

Figura 12 – Tela de Pontuação (Liga)

## **5.10 Tela de Atualização de Resultados**

Assim que o usuário clica no botão “Atualizar Resultados” da tela de informações do torneio, uma nova tela aparece para que o organizador desse torneio possa atualizar o resultado das partidas do torneio.

**Editar Torneio**

Partida Atual: jLabel1      Fulaninho X Beltranhinho

Numero do Jogo: 1       X

☐ Capote

Figura 13 – Tela de Atualização de Resultados (imagem do NetBeans)

## **6. Link do repositório do projeto**

[https://github.com/GMTonnera/trabalho\\_tp1](https://github.com/GMTonnera/trabalho_tp1)