

Trabalho Prático

Cartolitos CF

Élvis Júnior, 24/1038700
Gustavo Alves, 24/1020779
Pedro Marcinoni, 24/1002396
Grupo 1

¹Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0197 - Técnicas de Programação I

elvismirandajr@gmail.com, gusfring.a@gmail.com, pedroextrer@gmail.com

Abstract. This document describes the final project of the course "Técnicas de Programação I" at Universidade de Brasília. The project is a desktop application that allows users to manage their fantasy football teams, providing features such as team creation, player selection, and match simulation. The application is built following object-oriented programming principles and utilizes design patterns to ensure maintainability and scalability.

Resumo. Este documento descreve o projeto final da disciplina Técnicas de Programação I da Universidade de Brasília. O projeto é uma aplicação desktop que permite aos usuários gerenciar suas equipes de futebol fantasy, oferecendo recursos como criação de equipes, seleção de jogadores e simulação de partidas. A aplicação é construída seguindo princípios de programação orientada a objetos e utiliza padrões de design para garantir manutenibilidade e escalabilidade.

1. Descrição do Problema

Como bons brasileiros, quem não gosta de futebol? Foi pensando nisso que a emissora mais famosa do país, a Rede Globo, criou a Cartolitos CF, o novo Fantasy Game do momento, que reflete em tempo real os resultados estatísticos da Copa do Mundo de Clubes, o mais novo fenômeno do futebol. Veja também [?, ?].

Disponível para computador, o aplicativo permite que os usuários escalem seus times comprando jogadores dos 32 clubes do torneio com “cartoletas” — a moeda virtual do jogo — e pontue de acordo com o desempenho real desses atletas em campo na rodada. Assim, os participantes podem comparar seus resultados com os amigos em uma liga estilo tiro-curto, na qual se considera apenas uma única rodada.

Para colocar esse projeto em prática, a empresa designou três estudantes de Ciência da Computação — Elvis Miranda, Gustavo Alves e Pedro Marcinoni — para desenvolver um programa capaz de auxiliar na logística de gestão de qualquer liga de tiro-curto, servindo depois como base para o funcionamento integral do aplicativo.



Figura 1. Cartolitos CF - Fantasy Game

2. Definição das regras de negócio e divisão das tarefas

2.1. Regras de negócio

1. Função do programa:
 - (a) Cadastrar usuários, clubes, jogadores e equipes de usuários
 - (b) Simula estatísticas e pontuações conforme regras de negócios
 - (c) gera resultados da liga e rankings
2. Identificação: Cada usuário tem ID único, nome, e-mail e senha.
3. Orçamento: Cada usuário começa com 150 cartoletas.
4. Escalação:
 - (a) O time do usuário deve ter formação 4-3-3, ou seja, 1 goleiro, 4 defensores, 3 meio-campistas e 3 atacantes.
 - (b) O time do usuário não pode ter jogadores duplicados.
5. Capitão: Cada time deve ter um capitão, que recebe o dobro de pontos.
6. Estatísticas dos jogadores:
 - (a) Cada jogador possui um preço base e uma nota (overall) que varia de 0 a 100.
 - (b) Cada confronto (casa/fora) usa as notas de cada jogador para ajustar as probabilidades de contribuir com gols, assistências, desarmes, etc.
7. Pontuação:
 - (a) Goleiros:
 - Defesa de pênalti: +7 pontos
 - Defesa: +1.5 pontos
 - Gol sofrido: -1 ponto
 - Não sofrer gol: +5 pontos
 - (b) Defensores:
 - Desarme: +1.5 pontos
 - Falta cometida: -0.5 pontos
 - Gol contra: -5 pontos
 - Cartão amarelo: -1 pontos
 - Cartão vermelho: -3 pontos
 - (c) Meio-campistas:

- Assistência: +5 pontos
 - Gol: +8 pontos
 - Falta cometida: -0.5 pontos
 - Desarme: +1.5 pontos
- (d) Atacantes:
- Gol: +8 pontos
 - Assistência: +5 pontos
 - Finalização: +0.8 pontos
 - Falta cometida: -0.5 pontos
8. Simulação de partidas: O administrador do programa pode simular partidas entre os times cadastrados, gerando estatísticas e pontuações para cada jogador.
 9. Resultados da liga: O programa deve gerar resultados da liga, mostrando a pontuação total de cada time e o ranking dos usuários.

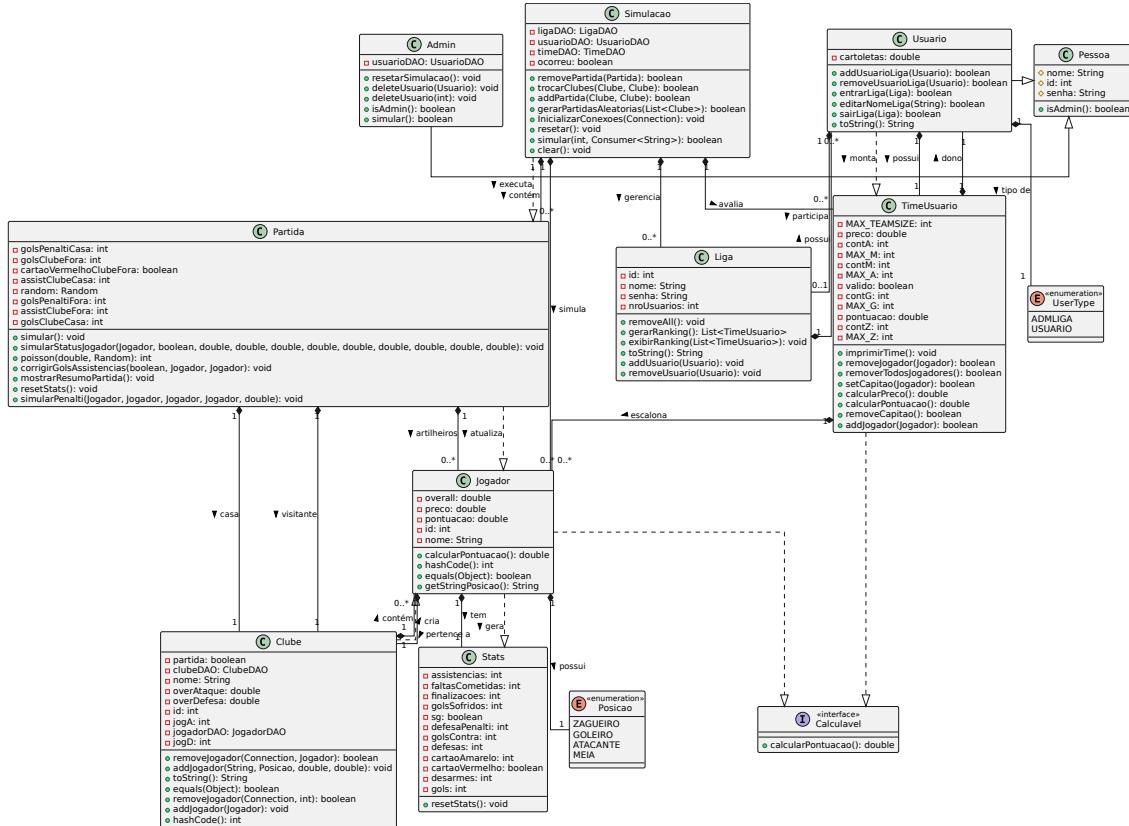
2.2. Divisão das Tarefas entre os Integrantes

A seguir está a divisão das tarefas realizada para o desenvolvimento do projeto final, buscando manter o equilíbrio entre as responsabilidades e funcionalidades implementadas por cada integrante:

- **Élvis Miranda**
 - Telas: Login/Cadastro de Usuário, Visualização de Jogadores, Edição de Perfil, Menu Principal do Usuário
 - Classes: Usuário, Jogador, Posição, Stats, autenticação
 - Funcionalidades: Cadastro, login, edição de perfil, consulta de jogadores, autenticação
 - Relatório: Descrição das telas de login, cadastro, visualização de jogadores e perfil; documentação das classes relacionadas
- **Pedro Marcinoni**
 - Telas: Montagem do Time, Clubes, Mercado, Edição de Clubes, Menu Principal do Administrador
 - Classes: TimeUsuario, Clube, manipulação de jogadores no time
 - Funcionalidades: Montagem e escalação do time, gerenciamento de clubes, mercado de jogadores
 - Relatório: Descrição das telas de montagem, clubes e mercado; documentação das classes relacionadas
- **Gustavo Alves**
 - Telas: Simulação de Partidas, Ranking da Liga, Gerenciamento de Ligas, Gerenciamento de Usuários, Gerenciamento de Partidas
 - Classes: Partida, Liga, Simulação, pontuação dos times
 - Funcionalidades: Simulação de partidas, geração de ranking, gerenciamento de ligas, usuários e partidas
 - Relatório: Descrição das telas de simulação, ranking, gerenciamento de ligas, usuários e partidas; documentação das classes relacionadas

Essa divisão permitiu que todos os integrantes participassem ativamente de diferentes etapas do desenvolvimento, garantindo uma contribuição equilibrada e colaborativa para o sucesso do projeto. Apesar de cada integrante ter focado em áreas específicas, todos colaboraram na integração das funcionalidades e na revisão do código, assegurando a coesão do sistema como um todo.

3. Diagrama de classes final



3.1. Classe abstrata Pessoa

A classe **Pessoa** é uma classe abstrata que representa uma pessoa genérica no sistema. Ela possui os seguintes **atributos**:

- `id`: um identificador único para cada pessoa.
- `nome`: armazena o nome da pessoa.
- `senha`: armazena a senha da pessoa para autenticação.

Além disso, a classe define os seguintes **métodos** (excluindo getters e setters):

- **Construtor Pessoa(int id, String nome, String senha)**: inicializa os atributos `id`, `nome` e `senha` ao criar uma nova instância da classe (ou de suas subclasses).
- **isAdmin()**: método que retorna `false` por padrão, indicando que a pessoa não é administradora. Esse método pode ser sobreescrito em subclasses para indicar se uma pessoa é administradora do sistema.

A classe serve como base para outras classes que representam tipos específicos de pessoas, permitindo reutilização e especialização de comportamento.

3.1.1. Subclasse Usuario

A classe **Usuario** é uma subclasse de **Pessoa** que representa um usuário do sistema, com funcionalidades específicas para participação em ligas e gerenciamento de seu time. Os principais **atributos** são:

- **cartoletas**: saldo virtual do usuário, utilizado para comprar jogadores (inicialmente 150).
- **tipo**: indica o tipo do usuário, podendo ser comum ou administrador de liga.
- **timeUsuario**: referência ao time montado pelo usuário.
- **liga**: referência à liga da qual o usuário faz parte.

Os principais **métodos** (excluindo getters e setters) são:

- **Usuario(int id, String nome, String senha, UserType tipo, Liga liga)**: construtor que inicializa o usuário com seus dados, tipo, liga e time.
- **editarNomeLiga(String novoNome)**: permite ao administrador de liga alterar o nome da liga.
- **addUsuarioLiga(Usuario usuario)**: permite ao administrador adicionar um novo usuário à liga.
- **removeUsuarioLiga(Usuario usuario)**: permite ao administrador remover um usuário da liga.
- **entrarLiga(Liga liga)**: permite ao usuário entrar em uma liga, caso ainda não participe de nenhuma.
- **sairLiga(Liga liga)**: permite ao usuário sair da liga atual; se for administrador de liga, seu tipo é alterado para comum.
- **toString()**: retorna uma representação textual do usuário, exibindo seu ID e nome.

Esses métodos garantem as principais operações de gerenciamento de ligas e participação dos usuários, respeitando as regras de negócio do sistema.

3.1.2. Subclasse Admin

A subclasse **Admin** representa o administrador do sistema, sendo uma especialização da classe abstrata **Pessoa**. Ela é responsável por operações administrativas, como gerenciamento de usuários e simulação de partidas. Os principais **atributos** são:

- **usuarioDAO**: objeto responsável pelo acesso e manipulação dos dados dos usuários no banco de dados.

Os principais **métodos** (excluindo getters e setters) são:

- **Admin(int id, String nome, String senha, Connection conn)**: construtor que inicializa o administrador e o objeto de acesso a dados.
- **simular()**: executa a simulação das partidas do sistema, passando por todas as etapas necessárias.
- **resetarSimulacao()**: reseta os dados da simulação realizada, restaurando o estado inicial.
- **deleteUsuario(Usuario usuario)**: remove um usuário do sistema a partir de um objeto **Usuario**.
- **deleteUsuario(int id)**: remove um usuário do sistema a partir do seu identificador.
- **isAdmin()**: sobrescreve o método da classe **Pessoa** para indicar que esta instância é de um administrador.

Esses métodos permitem ao administrador controlar usuários e simulações, garantindo a manutenção e o funcionamento correto do sistema.

3.2. Classe TimeUsuario

A classe **TimeUsuario** representa o time montado por um usuário, sendo responsável por gerenciar a escalação, o capitão, o cálculo de pontuação e o controle de validade do time para simulação.

- **Atributos principais:**

- `usuario`: referência ao usuário dono do time.
- `jogadores`: conjunto de jogadores escalados no time.
- `pontuacao`: pontuação total do time após a simulação.
- `preco`: valor total do time, somando o preço de todos os jogadores.
- `capitao`: jogador escolhido como capitão, que tem sua pontuação dobrada.
- `valido`: indica se a escalação está válida para simulação (time completo e capitão definido).
- `contG`, `contZ`, `contM`, `contA`: contadores de jogadores por posição (goleiro, zagueiro, meia e atacante).

- **Métodos principais:**

- `TimeUsuario(Usuario usuario)`: construtor que inicializa um time vazio para o usuário.
- `TimeUsuario(Usuario usuario, Set<Jogador> jogadores, Jogador jogcapitao)`: construtor que inicializa o time já com jogadores e capitão.
- `calcularPontuacao()`: calcula a pontuação total do time, considerando o capitão.
- `calcularPreco()`: calcula o preço total do time.
- `addJogador(Jogador jogador)`: adiciona um jogador ao time, respeitando as regras de quantidade e orçamento.
- `removeJogador(Jogador jogador)`: remove um jogador do time.
- `setCapitao(Jogador jogador)`: define o capitão do time, caso o jogador esteja escalado.
- `removeCapitao()`: remove o capitão do time.
- `removerTodosJogadores()`: remove todos os jogadores do time.
- `imprimirTime()`: imprime no console as informações do time, incluindo jogadores, pontuação e capitão.
- `getGoleiro()`, `getZagueiros()`, `getMeias()`, `getAtacantes()`: retornam os jogadores do time por posição.
- `getTodosOsEscalados()`: retorna todos os jogadores do time, organizados por posição.

Esses métodos permitem ao usuário montar, visualizar e gerenciar seu time, garantindo que a escalação siga as regras do sistema e esteja pronta para a simulação das partidas.

3.3. Classe Liga

A classe **Liga** representa uma liga do sistema, permitindo o agrupamento de usuários para competições e gerenciamento de rankings.

- **Atributos principais:**

- id: identificador único da liga.
- nroUsuarios: quantidade de usuários participantes da liga.
- nome: nome da liga.
- senha: senha de acesso à liga.
- usuarios: lista de usuários que participam da liga.

- **Métodos principais:**

- Liga(int id, String nome, String senha): construtor que inicializa a liga com identificador, nome e senha.
- removeAll(): remove todos os usuários da liga, fazendo com que cada usuário saia da liga.
- addUsuario(Usuario usuario): adiciona um usuário à liga e incrementa o número de participantes.
- removeUsuario(Usuario usuario): remove um usuário da liga e decrementa o número de participantes.
- gerarRanking(): gera e retorna uma lista dos times dos usuários, ordenada pela pontuação (ranking da liga).
- exibirRanking(List<TimeUsuario> times): exibe no console o ranking dos times da liga.
- toString(): retorna uma representação textual da liga, mostrando seu nome.

Esses métodos permitem o gerenciamento dos participantes e o acompanhamento do desempenho dos times dentro de uma liga, proporcionando a competição entre os usuários.

3.4. Classe Simulacao

A classe **Simulacao** é responsável por gerenciar o processo de simulação das partidas entre clubes, além de controlar o estado das simulações e o relacionamento com as ligas e usuários do sistema.

- **Atributos principais:**

- ocorreu: indica se a simulação já foi realizada.
- partidas: conjunto de partidas que serão simuladas.
- ligaDAO, usuarioDAO, timeDAO: objetos responsáveis pelo acesso e manipulação dos dados das ligas, usuários e times no banco de dados.
- ligasGlobal: lista global de ligas envolvidas na simulação.
- jogadoresSimuladosGlobal: mapa global de jogadores simulados, indexados por ID.
- timesComIdsGlobal: mapa global de times simulados, indexados por ID.

- **Métodos principais:**

- InicializarConexoes(Connection conn): inicializa os objetos de acesso a dados com a conexão ao banco.
- gerarPartidasAleatorias(List<Clube> clubes): sorteia clubes em partidas aleatórias, garantindo que todos participem.

- `addPartida(Clube clubeCasa, Clube clubeFora)`: adiciona uma partida entre dois clubes, se ambos ainda não estiverem em outra partida.
- `removePartida(Partida partida)`: remove uma partida da simulação.
- `simular(int etapa, Consumer<String> atualizarMensagem)`: executa a simulação das partidas em etapas, desde a verificação dos times até o salvamento dos resultados.
- `resetar()`: reseta o estado da simulação, restaurando os dados dos jogadores e times.
- `trocarClubes(Clube clube1, Clube clube2)`: troca clubes entre partidas diferentes.
- `clear()`: limpa todas as partidas da simulação e libera os clubes para novas partidas.

Esses métodos permitem o controle completo do ciclo de simulação das partidas, desde a geração dos confrontos até o cálculo e a reinicialização dos resultados, integrando-se com o banco de dados e garantindo a consistência das informações no sistema.

3.5. Interface Calculavel

A interface **Calculavel** define um contrato para as classes que desejam implementar a funcionalidade de cálculo de pontuação no sistema. Ela garante que qualquer classe que a implemente possua o método necessário para realizar esse cálculo.

- **Métodos principais:**

- `calcularPontuacao()`: método abstrato que deve ser implementado pelas classes concretas, responsável por calcular e retornar a pontuação de acordo com a lógica específica de cada classe.

Dessa forma, a interface **Calculavel** promove a padronização e a reutilização de código, permitindo que diferentes componentes do sistema possam calcular pontuações de maneira uniforme, como na classe `Jogador` e na classe `TimeUsuario`.

3.5.1. Classe Partida

A classe **Partida** representa um confronto entre dois clubes no sistema, sendo responsável por simular o jogo, calcular estatísticas dos jogadores e armazenar os principais eventos da partida.

- **Atributos principais:**

- `clubeCasa, clubeFora`: referências aos clubes que disputam a partida.
- `golsClubeCasa, golsClubeFora`: quantidade de gols marcados por cada clube.
- `assistClubeCasa, assistClubeFora`: quantidade de assistências de cada clube.
- `golsPenaltiCasa, golsPenaltiFora`: quantidade de gols de pênalti de cada clube.

- jogadoresGolCasa, jogadoresGolFora: listas de jogadores que marcaram gols por cada clube.
- jogadoresAssistenciaCasa, jogadoresAssistenciaFora: listas de jogadores que deram assistências por cada clube.
- cartaoVermelhoClubeCasa, cartaoVermelhoClubeFora: indicam se algum jogador do clube recebeu cartão vermelho.
- random: objeto para geração de números aleatórios, utilizado na simulação dos eventos da partida.

- **Métodos principais:**

- Partida(Clube clubeCasa, Clube clubeFora): construtor que inicializa uma partida entre dois clubes.
- simular(): executa toda a lógica de simulação da partida, gerando estatísticas para os jogadores e clubes.
- simularStatusJogador(...): simula os eventos individuais de cada jogador (gols, assistências, desarmes, faltas, cartões, etc.).
- simularPenalti(...): simula a ocorrência de pênaltis, gols e defesas de pênalti na partida.
- corrigirGolsAssistencias(...): ajusta a relação entre gols e assistências para garantir consistência nos dados da partida.
- resetStats(): reseta todas as estatísticas da partida e dos jogadores envolvidos, preparando para uma nova simulação.
- mostrarResumoPartida(): exibe no console um resumo detalhado dos principais eventos e estatísticas da partida.
- poisson(double lambda, Random random): calcula um valor aleatório baseado na distribuição de Poisson, utilizado para simular eventos como gols e assistências.
- calcularBonusClube(boolean timeCasa): calcula bônus de desempenho para o clube, considerando fatores como mando de campo e cartões vermelhos.
- getAllJogadores(): retorna uma lista com todos os jogadores participantes da partida.

Esses métodos e atributos permitem simular partidas de forma realista, atribuindo estatísticas individuais aos jogadores e resultados aos clubes, além de possibilitar o acompanhamento detalhado dos eventos ocorridos durante o jogo.

3.6. Classe Jogador

A classe **Jogador** representa um atleta disponível para ser escalado nos times dos usuários, armazenando informações essenciais para a simulação e pontuação.

- **Atributos principais:**

- id: identificador único do jogador.
- nome: nome do jogador.
- posicao: posição em que o jogador atua (goleiro, zagueiro, meia ou atacante).
- clube: referência ao clube ao qual o jogador pertence.
- preco: valor do jogador em cartoletas.

- overall: nota geral do jogador, utilizada para simulação de desempenho.
 - pontuacao: pontuação total do jogador após a simulação.
 - stats: objeto que armazena as estatísticas detalhadas do jogador na partida.
- **Métodos principais:**
 - Jogador(int id, String nome, Posicao posicao, Clube clube, double preco, double overall): construtor que inicializa o jogador com seus dados básicos.
 - getStringPosicao(): retorna a posição do jogador em formato textual.
 - calcularPontuacao(): calcula e atualiza a pontuação do jogador com base em suas estatísticas e regras de pontuação do sistema.
 - equals(Object obj) e hashCode(): métodos sobrescritos para garantir a correta comparação e uso do jogador em coleções.

Esses métodos e atributos permitem representar, identificar e calcular o desempenho de cada jogador, sendo fundamentais para a lógica de escalação e simulação das partidas.

3.7. Classe Stats

A classe **Stats** representa as estatísticas detalhadas de desempenho de um jogador em uma partida, armazenando informações essenciais para o cálculo da pontuação individual.

- **Atributos principais:**
 - desarmes: quantidade de desarmes realizados pelo jogador.
 - gols: quantidade de gols marcados.
 - assistencias: quantidade de assistências realizadas.
 - sg: indica se o jogador terminou a partida sem sofrer gols (aplicável a goleiros e zagueiros).
 - finalizacoes: número de finalizações feitas.
 - defesas: número de defesas realizadas (goleiros).
 - defesaPenalti: número de defesas de pênalti.
 - golsContra: quantidade de gols contra marcados.
 - cartaoVermelho: indica se o jogador recebeu cartão vermelho.
 - golsSofridos: quantidade de gols sofridos (goleiros e zagueiros).
 - cartaoAmarelo: quantidade de cartões amarelos recebidos.
 - faltasCometidas: número de faltas cometidas.
 - posicao: posição do jogador na partida.
- **Métodos principais:**
 - resetStats(): reseta todas as estatísticas do jogador para os valores iniciais, preparando para uma nova simulação.
 - forEach(Consumer<? super Map.Entry<String, Integer>> action): percorre todas as estatísticas, permitindo executar uma ação para cada uma delas (útil para exibição ou processamento).

Esses métodos e atributos permitem registrar, manipular e exibir as estatísticas individuais dos jogadores, sendo fundamentais para o cálculo da pontuação e para a análise de desempenho.

3.8. Classe Clube

A classe **Clube** representa um clube de futebol no sistema, armazenando informações sobre seus jogadores, desempenho médio e integração com o banco de dados.

- **Atributos principais:**

- `id`: identificador único do clube.
- `nome`: nome do clube.
- `overAtaque`: média aritmética do overall dos jogadores de ataque do clube.
- `overDefesa`: média aritmética do overall dos jogadores de defesa do clube.
- `jogadores`: conjunto de jogadores pertencentes ao clube.
- `jogA`, `jogD`: quantidade de jogadores de ataque e defesa, respectivamente.
- `partida`: indica se o clube já está escalado para uma partida.
- `clubeDAO`, `jogadorDAO`: objetos responsáveis pelo acesso e manipulação dos dados do clube e dos jogadores no banco de dados.

- **Métodos principais:**

- `Clube(...)`: construtores que inicializam o clube, podendo inserir ou recuperar dados do banco de dados.
- `addJogador(String nomeJogador, Posicao posicao, double preco, double overall)`: adiciona um novo jogador ao clube e ao banco de dados.
- `addJogador(Jogador jogador)`: adiciona ao clube um jogador já existente no banco de dados.
- `removeJogador(Connection conn, Jogador jogador)`: remove um jogador do clube e do banco de dados, se a simulação ainda não ocorreu.
- `removeJogadorById(Connection conn, int idJogador)`: remove um jogador pelo ID, tanto do clube quanto do banco de dados.
- `recalcOverAtaqueAdd/Sub(double overall)` e `recalcOverDefesaAdd/Sub(double overall)`: recalculam a média de overall de ataque ou defesa ao adicionar ou remover jogadores.
- `isAtaque(Posicao posicao)` e `isDefesa(Posicao posicao)`: verificam se uma posição é considerada de ataque ou defesa.
- `toString()`: retorna uma representação textual do clube.
- `equals(Object o)` e `hashCode()`: métodos sobrescritos para garantir a correta comparação e uso do clube em coleções.

Esses métodos e atributos permitem o gerenciamento completo dos clubes, incluindo a manutenção dos jogadores, atualização das médias de desempenho e integração com o banco de dados do sistema.

4. Telas

4.1. Telas Iniciais

4.1.1. Tela de Início

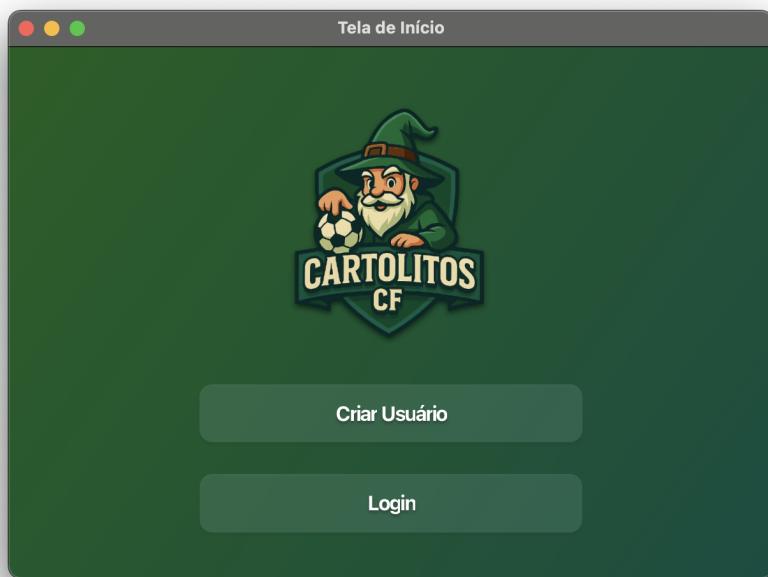


Figura 2. Tela de Início do Cartolitos CF

A tela de início é a primeira interface que o usuário vê ao abrir o programa. Ela apresenta o logotipo do Cartolitos CF e opções para acessar as funcionalidades de login e de cadastro.

4.1.2. Tela de Login

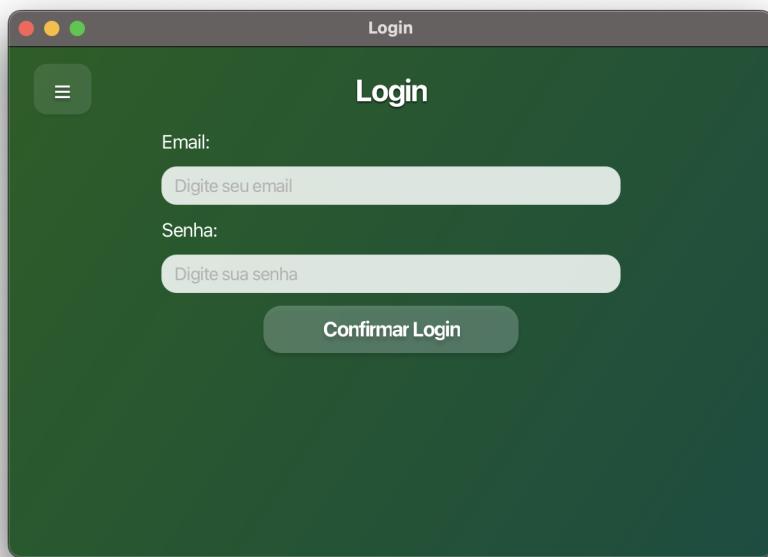


Figura 3. Tela de Login do Cartolitos CF

A tela de login permite que o usuário insira suas credenciais (nome e senha) para acessar sua conta. Caso o usuário não tenha uma conta, ele pode optar por se cadastrar.

4.1.3. Tela de Cadastro



Figura 4. Tela de Cadastro do Cartolitos CF

A tela de cadastro permite que novos usuários criem uma conta no sistema, fornecendo um nome, e-mail e senha. Após o cadastro, o usuário é redirecionado para a tela de login.

O programa possui dois fluxos principais: o fluxo do usuário e o fluxo do administrador. Cada um possui telas específicas para suas funcionalidades.

4.2. Fluxo do usuário

4.2.1. Menu Principal



Figura 5. Menu Principal do Cartolitos CF

A tela do menu principal é exibida após o login bem-sucedido do usuário. Ela apresenta as opções disponíveis para o usuário, como visualizar seu time, participar de ligas, escalar jogadores e editar seu perfil. O usuário pode navegar entre essas opções para acessar as funcionalidades do sistema.

4.2.2. Ligas

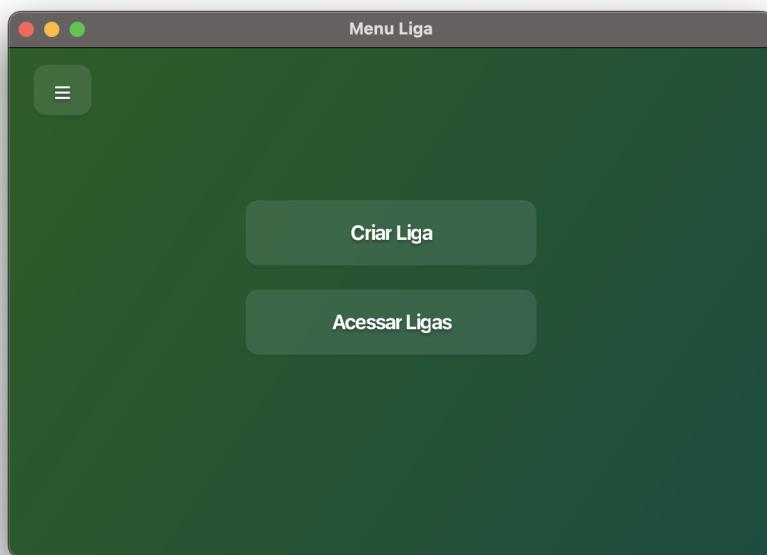


Figura 6. Tela de Ligas do Cartolitos CF



Figura 7. Tela de Acesso às Ligas do Cartolitos CF

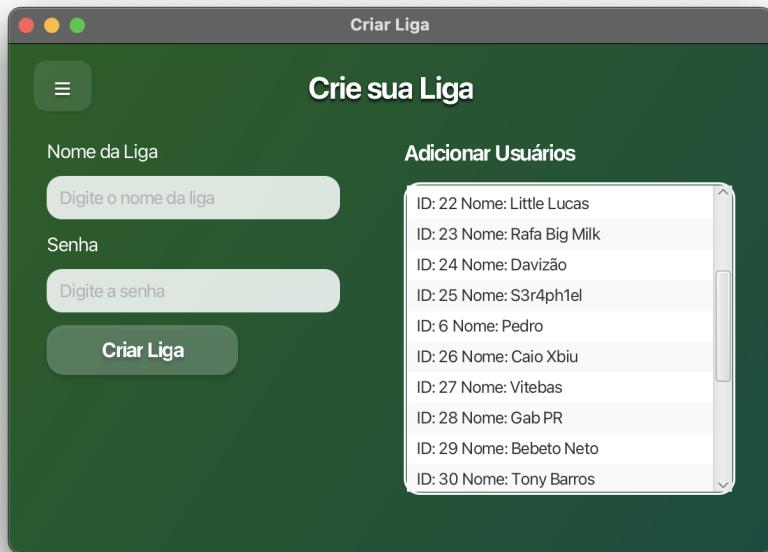


Figura 8. Tela de Criação de Liga do Cartolitos CF

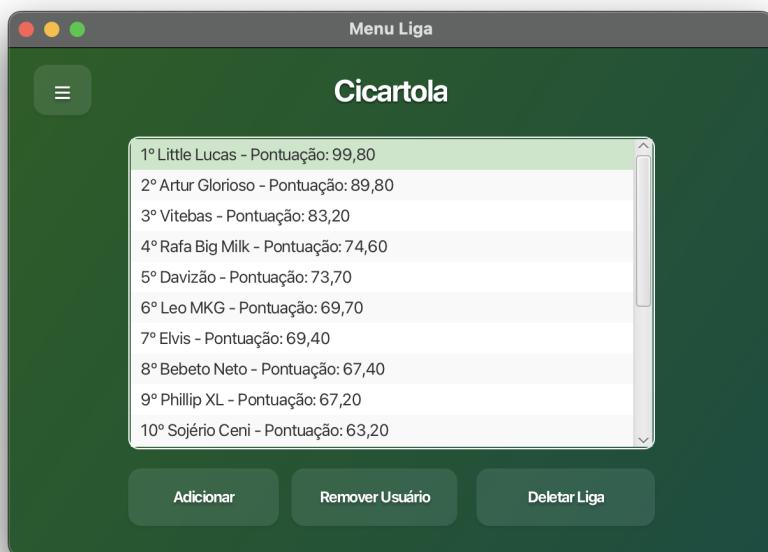


Figura 9. Tela de Detalhes da Liga do Cartolitos CF

Quando o usuário acessa a opção de ligas, ele é direcionado para uma tela que exibe as opções relacionadas às ligas. Se o usuário ainda não estiver em uma liga, ele pode optar por entrar em uma liga existente ou criar uma nova liga. Caso já esteja em uma liga, ele pode visualizar os detalhes da liga, como participantes e pontuações (caso as partidas já

tenham sido simuladas).

4.2.3. Escalar Jogadores



Figura 10. Tela de Escalação de Jogadores do Cartolitos CF

The screenshot shows the 'Mercado' interface. The title 'Mercado' is at the top. Below it is a search bar with 'Todos' selected and a 'Pesquisar jogador pelo nome' input field. A 'Pesquisar' button is also present. The main area is a table with columns: TIME, POSIÇÃO, JOGADOR, PREÇO, and COMPRAR?. The table lists players from Flamengo, categorized by position: ATACANTE, ATACANTE, ATACANTE, MEIA, MEIA, MEIA, and ZAGUEIRO. For each player, there is a 'Comprar' (Buy) button in a green box and a 'Remover' (Remove) button in a red box.

TIME	POSIÇÃO	JOGADOR	PREÇO	COMPRAR?
Flamengo	ATACANTE	Pedro	C\$15,00	<button>Remover</button>
Flamengo	ATACANTE	Bruno Henrique	C\$12,00	<button>Comprar</button>
Flamengo	ATACANTE	Luiz Araujo	C\$11,00	<button>Comprar</button>
Flamengo	MEIA	Arrascaeta	C\$12,00	<button>Remover</button>
Flamengo	MEIA	De la Cruz	C\$12,00	<button>Comprar</button>
Flamengo	MEIA	Jorginho	C\$14,00	<button>Comprar</button>
Flamengo	ZAGUEIRO	Alex Sandro	C\$14,00	<button>Comprar</button>

Figura 11. Tela de Mercado do Cartolitos CF

A tela de escalação de jogadores permite que o usuário monte seu time, escolhendo jogadores disponíveis no mercado. O usuário pode visualizar os jogadores por posição

(goleiro, zagueiro, meio-campista e atacante) e selecioná-los para compor sua equipe. A tela também exibe informações sobre nome, clube, preço.

4.2.4. Partidas

Mandante		Visitante
Borussia Dortmund	Detalhes	Bayern München
Boca Juniors	Detalhes	Benfica
Chelsea	Detalhes	Ulsan HD
RB Salzburg	Detalhes	Pachuca
Wydad AC	Detalhes	Internazionale
Seattle Sounders	Detalhes	Palmeiras
Flamengo	Detalhes	Manchester City
Fluminense	Detalhes	Atlético Madrid
Los Angeles FC	Detalhes	Al Hilal
Auckland City	Detalhes	Paris Saint-German
River Plate	Detalhes	Inter Miami
Urawa Reds	Detalhes	Al Ain
Botafogo	Detalhes	Esperânce Tunis
Juventus	Detalhes	Monterrey
Real Madrid	Detalhes	Mamelodi Sundowns
FC Porto	Detalhes	Al Ahly

Figura 12. Tela de Partidas do Cartolitos CF

DETALHES DA PARTIDA

Flamengo 1 x 2 Palmeiras

PONTUAÇÃO TOTAL 48,40 x 54,60

Flamengo	Palmeiras	Jogador	Pontuação	
POS				
ATACANTE	Pedro		2,90	Ver detalhes
ATACANTE	Bruno Henrique		1,30	Ver detalhes
ATACANTE	Luiz Araújo		3,40	Ver detalhes
MEIA	Arrascaeta		9,80	Ver detalhes
MEIA	De la Cruz		4,10	Ver detalhes
MEIA	Jorginho		11,10	Ver detalhes
ZAGUEIRO	Alex Sandro		-0,20	Ver detalhes
ZAGUEIRO	Léo Ortiz		3,00	Ver detalhes
ZAGUEIRO	Léo Pereira		4,80	Ver detalhes
ZAGUEIRO	Wesley		1,80	Ver detalhes
GOLEIRO	A.Rossi		4,00	Ver detalhes
ATACANTE	Cebolinha		2,40	Ver detalhes

Figura 13. Tela de Partidas Simuladas do Cartolitos CF

A tela de partidas exibe as partidas simuladas entre os clubes da liga. O usuário pode visualizar os resultados das partidas, incluindo gols, assistências e estatísticas dos jogadores. Essa tela permite que o usuário acompanhe o desempenho de seu time e dos demais participantes da liga.

4.2.5. Jogadores

Jogadores

Todos

Pesquisar jogador pelo nome

NOME	POSIÇÃO	CLUBE	PONTUAÇÃO ▾	DETALHES
Rodri	MEIA	Manchester City	29,00	Detalhes
Bidstrup	MEIA	RB Salzburg	21,60	Detalhes
Borja	ATACANTE	River Plate	19,10	Detalhes
Lautaro Martínez	ATACANTE	Internazionale	17,40	Detalhes
Cebolinha	ATACANTE	Flamengo	17,10	Detalhes
João Cancelo	ZAGUEIRO	Al Hilal	16,80	Detalhes
Mbappé	ATACANTE	Real Madrid	16,20	Detalhes
Kaku	MEIA	Al Ain	15,90	Detalhes
Savarino	MEIA	Botafogo	15,10	Detalhes
Vini Jr	ATACANTE	Real Madrid	15,00	Detalhes

Figura 14. Tela de Jogadores do Cartolitos CF



Figura 15. Tela de Detalhes do Jogador do Cartolitos CF

A tela de jogadores permite que o usuário visualize todos os jogadores disponíveis no sistema, incluindo aqueles que ele já escalou em seu time. O usuário pode acessar detalhes de cada jogador, como estatísticas, pontuação e preço. Essa tela é útil para o usuário acompanhar o desempenho dos jogadores e tomar decisões sobre escalação.

4.2.6. Editar Perfil do Usuário

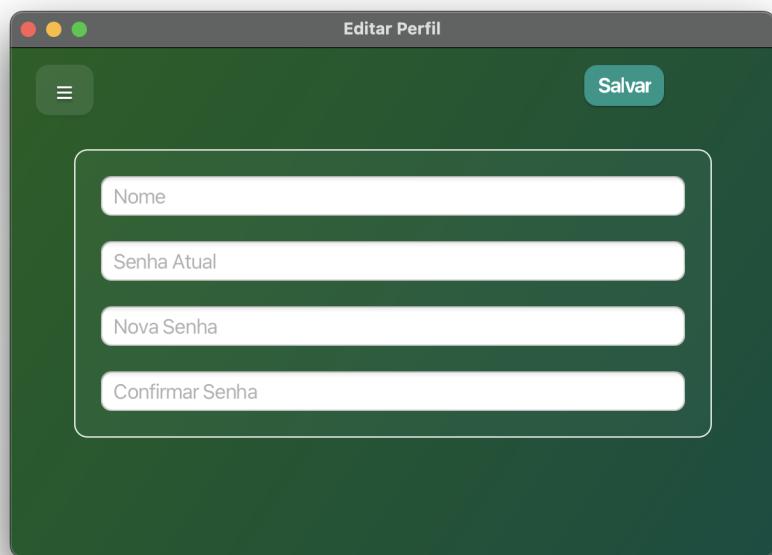


Figura 16. Tela de Edição de Perfil do Usuário do Cartolitos CF

A tela de edição de perfil permite que o usuário altere suas informações pessoais, como nome e senha. O usuário pode atualizar seus dados e salvar as alterações para manter seu perfil atualizado.

4.3. Fluxo do administrador

4.3.1. Menu do Administrador

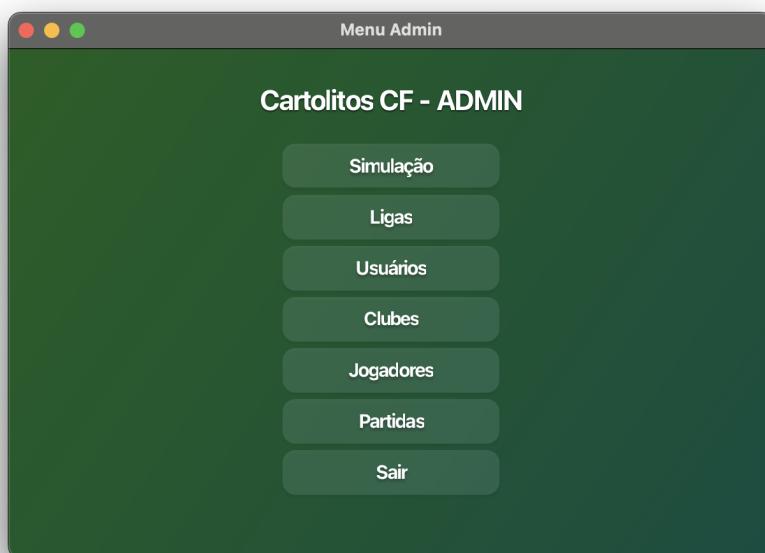


Figura 17. Menu do Administrador do Cartolitos CF

A tela do menu do administrador é exibida após o login bem-sucedido de um usuário com privilégios de administrador. Ela apresenta as opções disponíveis para o administrador, como simular partidas, gerenciar usuários e ligas, editar clubes, jogadores e confrontos entre clubes. O administrador pode navegar entre essas opções para acessar as funcionalidades administrativas do sistema.

4.3.2. Simular Partidas

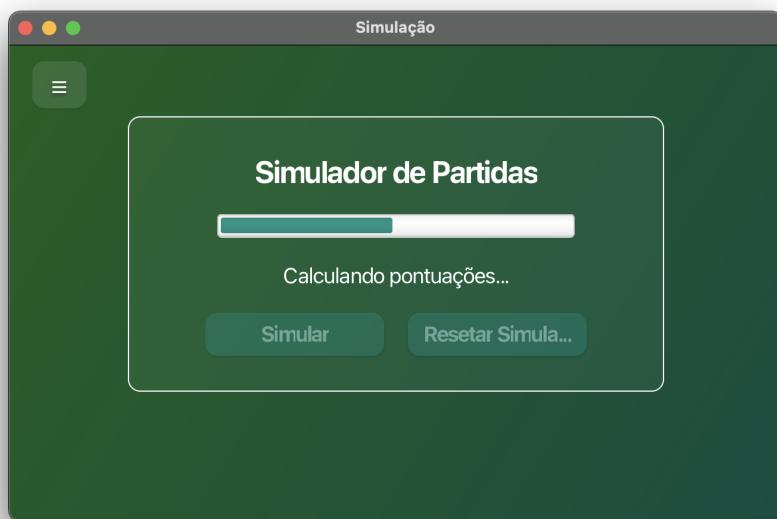


Figura 18. Tela de Simulação de Partidas do Cartolitos CF

A tela de simulação de partidas permite que o administrador execute a simulação das partidas entre os clubes cadastrados. O administrador pode iniciar a simulação, que irá gerar estatísticas e pontuações para cada jogador, além de atualizar os resultados da liga.

4.3.3. Gerenciar Ligas

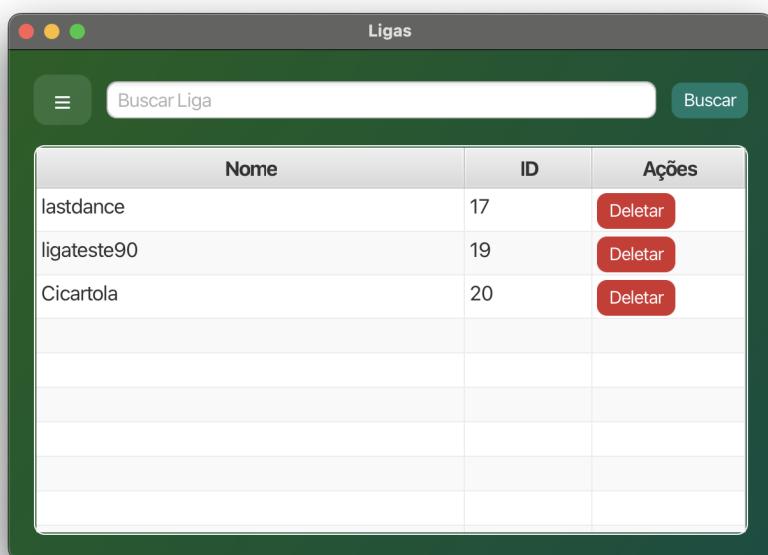


Figura 19. Tela de Gerenciamento de Ligas do Cartolitos CF

A tela de gerenciamento de ligas permite que o administrador visualize todas as ligas cadastradas no sistema. O administrador pode acessar os detalhes de cada liga, editar informações, adicionar ou remover usuários e deletar ligas, se necessário. Essa tela é essencial para o controle das competições e para garantir que as ligas estejam funcionando corretamente.

4.3.4. Gerenciar Usuários

Nome	Tipo	ID	Ações
Elvis	ADM LIGA	9	<button>Deletar</button>
FD Braguinha	USUARIO	19	<button>Deletar</button>
Eric China	USUARIO	20	<button>Deletar</button>
Phillip XL	USUARIO	21	<button>Deletar</button>
Little Lucas	USUARIO	22	<button>Deletar</button>
Rafa Big Milk	USUARIO	23	<button>Deletar</button>
Davizão	USUARIO	24	<button>Deletar</button>
Caio Xbiu	USUARIO	26	<button>Deletar</button>
Vitebas	USUARIO	27	<button>Deletar</button>
Gab PR	USUARIO	28	<button>Deletar</button>
Bebeto Neto	USUARIO	29	<button>Deletar</button>

Figura 20. Tela de Gerenciamento de Usuários do Cartolitos CF

A tela de gerenciamento de usuários permite que o administrador visualize todos os usuários cadastrados no sistema. O administrador pode acessar os detalhes de cada usuário e excluir usuários, se necessário. Essa tela é importante para manter o controle sobre os participantes do sistema e garantir que apenas usuários válidos tenham acesso às funcionalidades.

4.3.5. Gerenciar Clubes

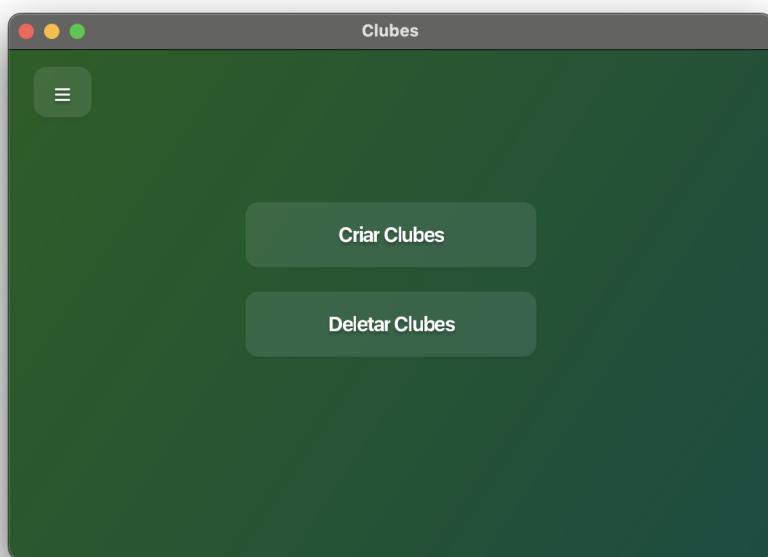


Figura 21. Tela de Gerenciamento de Clubes do Cartolitos CF



Figura 22. Tela de Edição de Clubes do Cartolitos CF

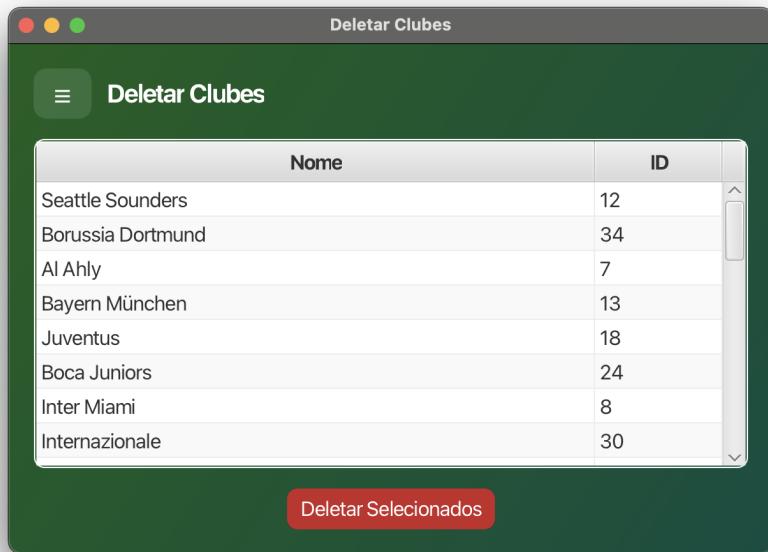


Figura 23. Tela de Edição de Clubes do Cartolitos CF

A tela de gerenciamento de clubes permite que o administrador visualize todos os clubes cadastrados no sistema. O administrador pode acessar os detalhes de cada clube, adicionar dois novos clubes (pois não pode haver um clube sem jogo na rodada) e excluir clubes, se necessário. Essa tela é essencial para manter o controle sobre os clubes participantes do sistema e garantir que as informações estejam atualizadas.

4.3.6. Gerenciar Jogadores



Figura 24. Tela de Gerenciamento de Jogadores do Cartolitos CF



Figura 25. Tela de Edição de Jogadores do Cartolitos CF

The screenshot shows a dark-themed application window titled "Deletar Jogadores". At the top, there is a search bar with placeholder text "Buscar por nome ou clube," and a blue "Buscar" button. Below the header, the title "Jogadores Cadastrados" is centered. A table follows, with columns labeled "Nome", "Clube", "OVR", "Preço", and "Ações". The "Ações" column contains red "Deletar" buttons next to each player's name. The data in the table is as follows:

Nome	Clube	OVR	Preço	Ações
A. Dari	Al Ahly	71.0	4.0	<button>Deletar</button>
Y. Ibrahim	Al Ahly	68.0	4.0	<button>Deletar</button>
M. Hany	Al Ahly	70.0	4.0	<button>Deletar</button>
El Shenawy	Al Ahly	72.0	7.0	<button>Deletar</button>
Benchmarki	Al Ahly	73.0	8.0	<button>Deletar</button>
Trézéguet	Al Ahly	75.0	9.0	<button>Deletar</button>
W. Ali	Al Ahly	72.0	10.0	<button>Deletar</button>
Zizo	Al Ahly	75.0	6.0	<button>Deletar</button>
E. Ashour	Al Ahly	74.0	9.0	<button>Deletar</button>
Ben Romdhane	Al Ahly	75.0	8.0	<button>Deletar</button>
A. Dieng	Al Ahly	71.0	6.0	<button>Deletar</button>

Figura 26. Tela de Edição de Jogadores do Cartolitos CF

A tela de gerenciamento de jogadores permite que o administrador visualize todos os jogadores cadastrados no sistema. O administrador pode acessar os detalhes de cada jogador, adicionar novos jogadores e excluir jogadores, se necessário. Essa tela é importante para manter o controle sobre os atletas disponíveis no sistema e garantir que as informações estejam atualizadas.

4.3.7. Gerenciar Partidas

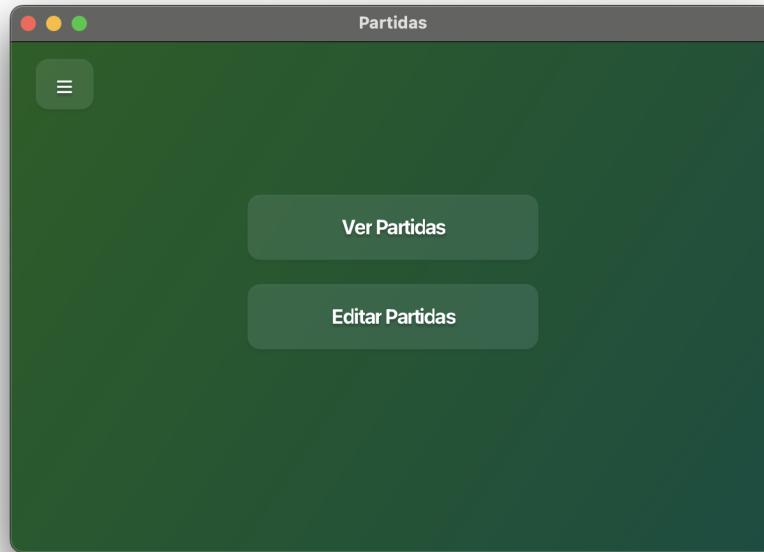


Figura 27. Tela de Gerenciamento de Partidas do Cartolitos CF



Figura 28. Tela de Edição de Partidas do Cartolitos CF

A tela de gerenciamento de partidas permite que o administrador visualize todas as partidas simuladas entre os clubes. O administrador pode acessar os detalhes de cada partida,

editar os confrontos, se necessário. Essa tela é essencial para manter o controle sobre os confrontos realizados no sistema e garantir que as informações estejam atualizadas.

5. Conclusão

O Cartolitos CF é um sistema de gerenciamento de ligas de futebol virtual, inspirado no famoso Cartola FC e no Rei do Pitaco. Ele permite que usuários criem e gerenciem seus times, escalem jogadores, participem de ligas e simulem partidas, tudo isso com uma interface amigável e intuitiva. O sistema foi desenvolvido utilizando Java, com uma arquitetura orientada a objetos que facilita a manutenção e a expansão das funcionalidades, conforme abordado em obras como [?, ?, ?]. As principais classes do sistema, como Usuario, Administrador, TimeUsuario, Liga, Simulacao, Jogador, Stats e Clube, foram projetadas para interagir de forma coesa, permitindo uma experiência fluida para o usuário, seguindo princípios de orientação a objetos descritos em [?, ?]. A implementação do sistema também inclui uma interface gráfica desenvolvida com JavaFX e um banco de dados PostgreSQL para persistência de dados, garantindo que as informações dos usuários, times, ligas e partidas sejam armazenadas de forma segura e eficiente.

Referências

- [Cartola FC] Cartola FC. Wikipédia. https://pt.wikipedia.org/wiki/Cartola_FC. [Online; acesso em 01-jun-2025].
- [Esporte fantasy] Esporte fantasy. Wikipédia. https://pt.wikipedia.org/wiki/Esporte_fantasy. [Online; acesso em 01-jun-2025].
- [et al 2007] et al, G. B. (2007). *Object-Oriented Analysis and Design with Applications*. Addison-Wesley Professional, 3rd edition.
- [Horstmann and Cornell 2012] Horstmann, C. S. and Cornell, G. (2012). *Core Java, Volume I - Fundamentals*. Prentice Hall, 9th edition.
- [Horstmann and Cornell 2013] Horstmann, C. S. and Cornell, G. (2013). *Core Java, Volume II - Advanced Features*. Prentice Hall, 9th edition.
- [Meyer 1997] Meyer, B. (1997). *Object-Oriented Software Construction*. Prentice Hall, 2nd edition.
- [Weisfeld 2013] Weisfeld, M. (2013). *The Object-Oriented Thought Process*. Addison-Wesley, 4th edition.