BANCO DE DADOS

- Alexandre Dantas
- Bergony Silva
- Marlus Silva

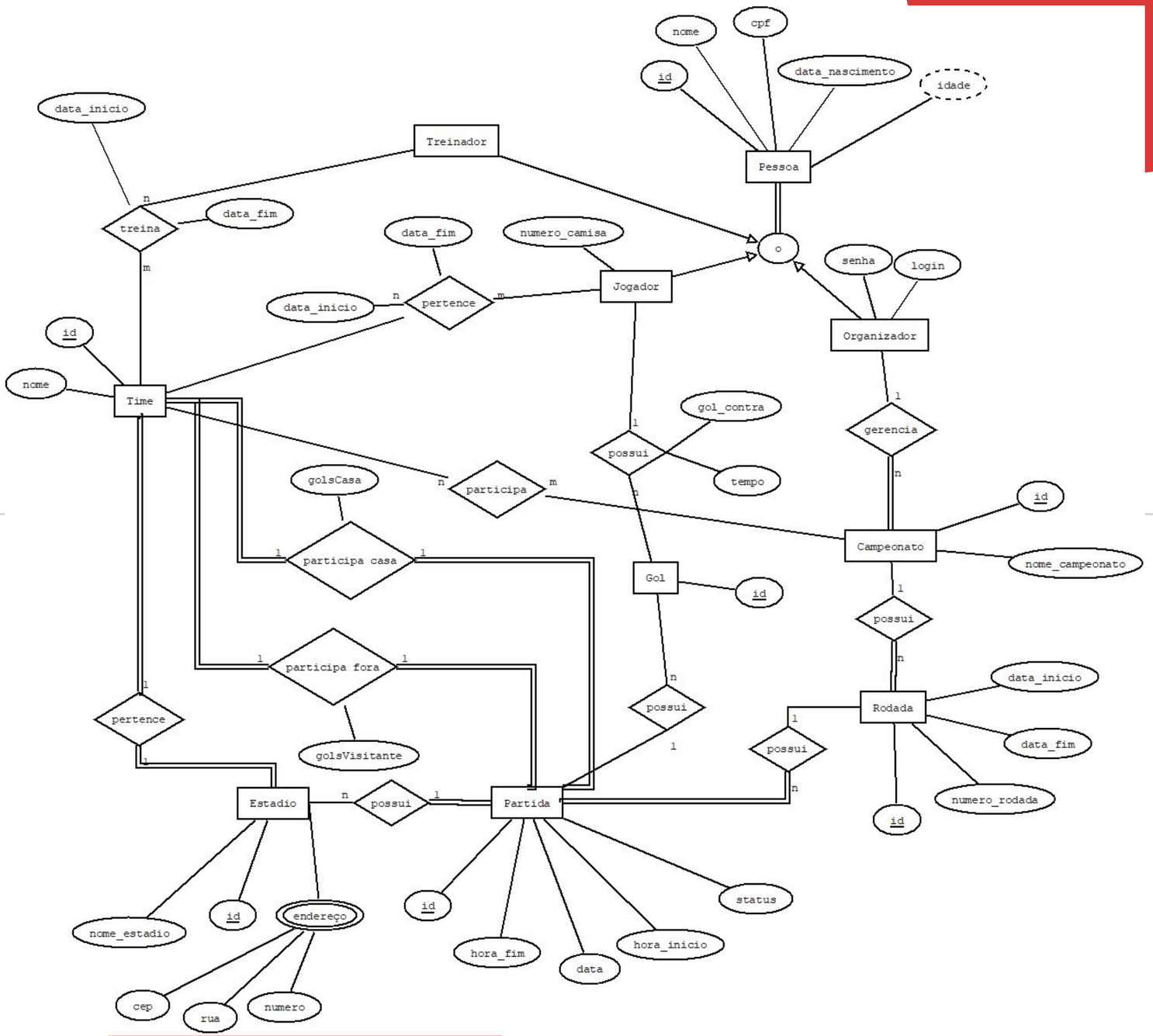
Descrição

O objetivo deste projeto é fazer um sistema com banco de dados capaz de fazer o gerenciamento de campeonato de times de futebol.

Motivação: Grupo apaixonado por futebol.

EER

- 2 SEMANAS PARA DESENVOLVIMENTO
- 10 ENTIDADES
- 12 RELACIONAMENTOS



Modelo Relacional

CRIAÇÃO DAS ENTIDADES

Criação das entidades dispostas no diagrama ER.

CRIAR RELAÇÕES

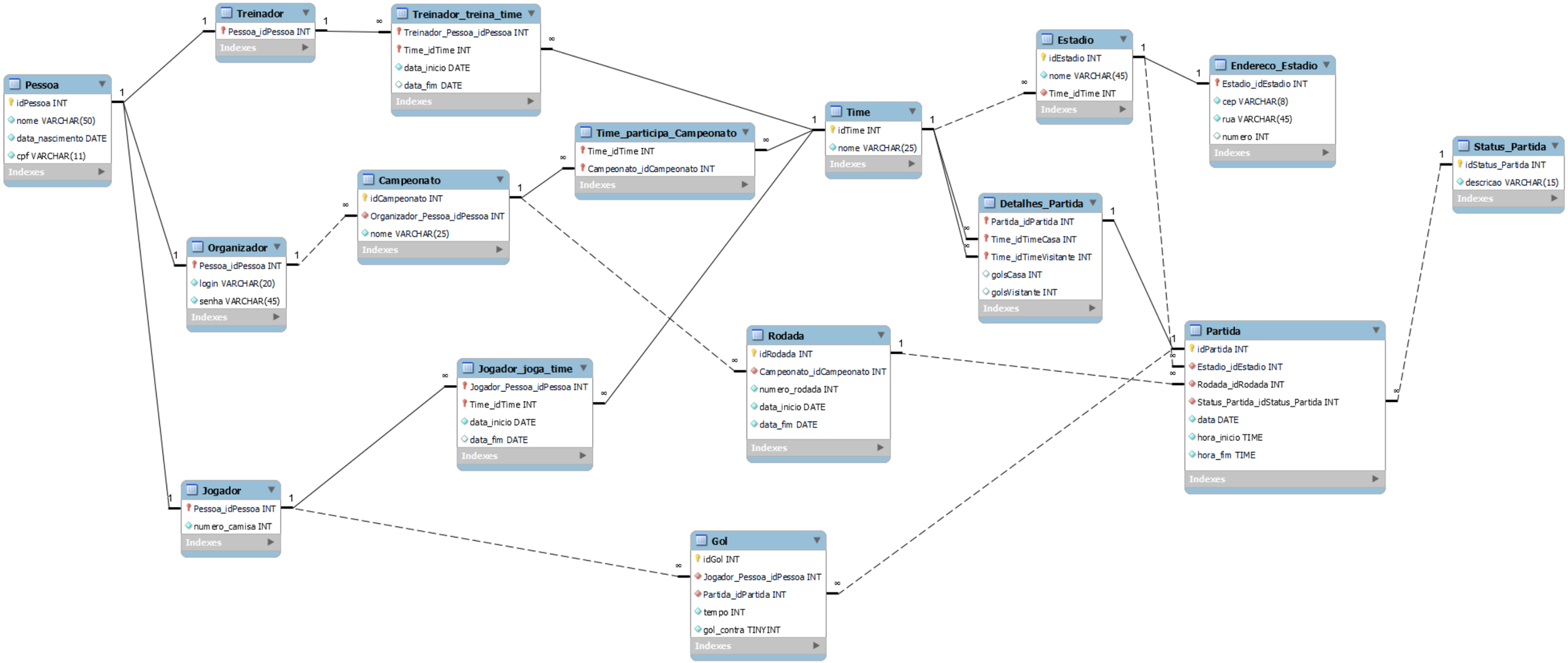
Criação das relações onde existem herança: Pessoa, Jogador, Treinador e Organizador.

CRIAR RELACIONAMENTOS

Neste ponto, criação dos relacionamentos 1:N.

RELACIONAMENTOS FINAIS

Adição dos relacionamentos: N:M (criação de novas entidades)



16 Tabelas

2 semanas para desenvolvimento

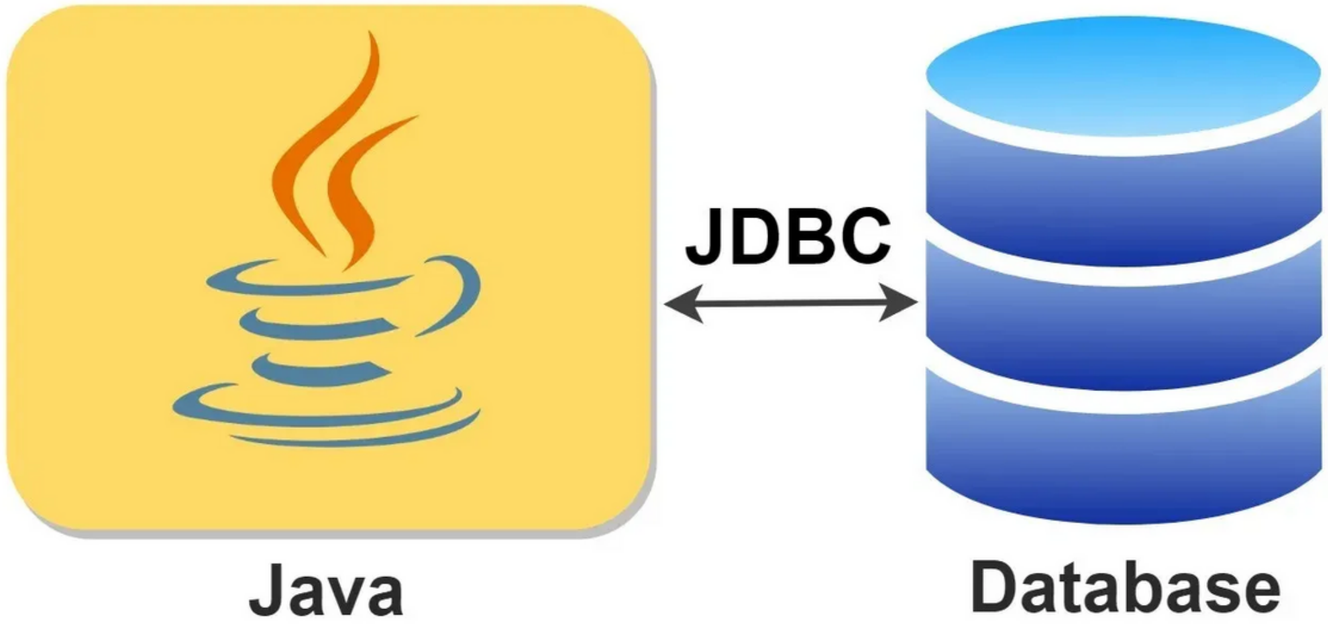
- Ajuste na tabela Estadio - Criação da tabela Endereco_Estadio.
- Ajuste na tabela Partida - Criação da tabela Status_Partida

Normalização

Processo de Normalização do Banco de Dados

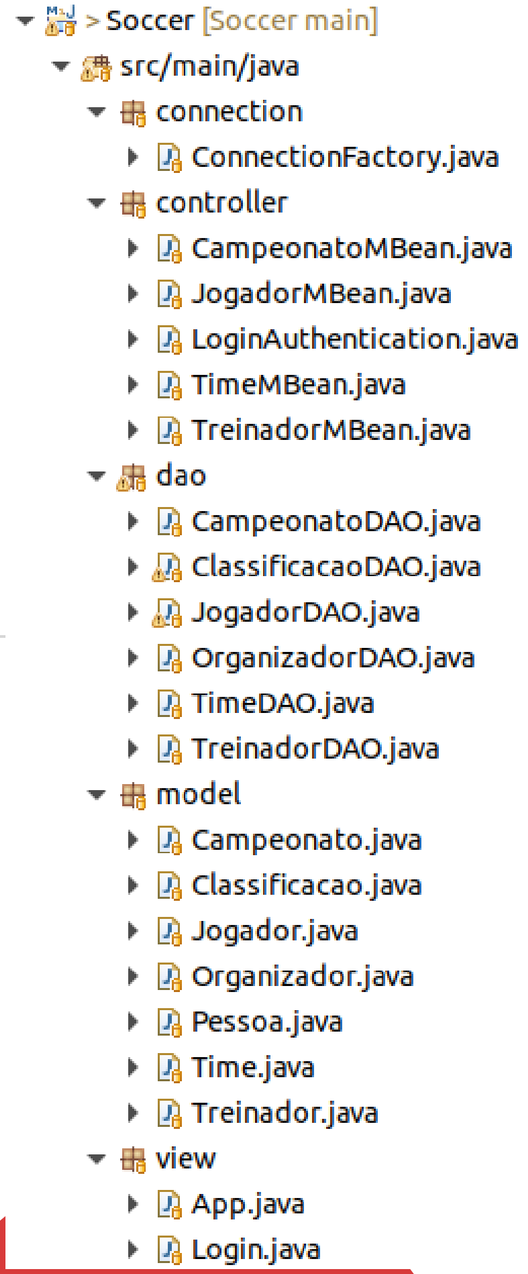
Implementação

Tecnologias utilizadas



Estrutura

- 21 CLASSES
- JAVA 17
- JAVA JDBC
- MYSQL 8.0.33



Execução

```
------------------------------------------------------------------
Bem vindo ao sistema Soccer.
Escolha a opção desejada:
1 - Realizar Login
0 - Sair
Opção: 1
Informe o login: organizador1
Informe a senha: senhaorganizador1
Usuário: organizador1 autenticado com sucesso!

Informe a opção desejada:
1 - Consultar Jogador
2 - Consultar Treinador
3 - Consultar Time
4 - Média de gols do campeonato
5 - Artilheiros do campeonato
6 - Consultar Classificação
7 - Consultar Histórico Atleta
8 - Jogadores por time
9 - Atualizar Jogador
10 - Cadastrar Jogador
0 - Sair
Opção:
```

Connection Factory

```java
import java.sql.Connection;

public class ConnectionFactory {

    private static final String URL = "jdbc:mysql://localhost:3306/mydb?useSSL=false";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "root";

    public static Connection getConnection() {
        try {
            return DriverManager.getConnection(URL, USERNAME, PASSWORD);
        } catch (SQLException e) {
            System.out.println("Erro ao conectar ao banco de dados: " + e.getMessage());
            return null;
        }
    }
}
```

Model Jogador

```
private static void realizarConsultaAtleta(String nome, String equipe) {

    Jogador jogador = new Jogador();
    JogadorDAO jogadorDAO = new JogadorDAO();

    jogador = jogadorDAO.getJogadorByNomeTime(nome, equipe);

    if (jogador != null) {
        System.out.println("Dados localizados com sucesso.");
        System.out.println("Nome do Jogador: " + jogador.getNome());
        System.out.println("Data de Nascimento: " + jogador.getDataNascimento());
        System.out.println("CPF: " + jogador.getCpf());
        System.out.println("Equipe Atual: " + jogador.getTimeAtual());
        System.out.println("");
    }

    else {
        System.out.println("Não foram encontrados atletas com nome e equipe informadas.");
    }
}
```

Consulta Média de Gols

```java
public class CampeonatoDAO {

    public Double getMediaGolsCampeonato(String nome) {
        String sql = "SELECT AVG(total_gols) as media " + "FROM ( "
                + "SELECT SUM(golsCasa + golsVisitante) as total_gols " + "FROM Detalhes_Partida dp "
                + "INNER JOIN Partida p ON dp.Partida_idPartida = p.idPartida "
                + "INNER JOIN Rodada r ON p.idPartida = r.idRodada "
                + "INNER JOIN Campeonato c ON r.Campeonato_idCampeonato = c.idCampeonato " + "WHERE c.nome = ? "
                + "GROUP BY Partida_idPartida " + ") subquery";

        try {

            Connection conn = ConnectionFactory.getConnection();
            java.sql.PreparedStatement pstm = conn.prepareStatement(sql);

            // Definindo os valores dos parâmetros
            pstm.setString(1, nome); // nome

            // Executando o comando SQL
            ResultSet resultSet = pstm.executeQuery();

            // Iterando sobre os resultados da consulta
            while (resultSet.next()) {
                Double media = resultSet.getDouble("media");

                return media;
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }
```

Consulta Classificação

```
public static List<Classificacao> getClassificacao(String nome) {

    String sql = "SELECT t.idTime, t.nome AS Time, " + "COALESCE(SUM(CASE WHEN dp.Time_idTimeCasa = t.idTime THEN "
            + "CASE WHEN dp.golsCasa > dp.golsVisitante THEN 3 " + "WHEN dp.golsCasa = dp.golsVisitante THEN 1 "
            + "ELSE 0 END " + "ELSE " + "CASE WHEN dp.golsCasa < dp.golsVisitante THEN 3 "
            + "WHEN dp.golsCasa = dp.golsVisitante THEN 1 " + "ELSE 0 END " + "END), 0) AS Pontos, "
            + "COALESCE(SUM(CASE WHEN dp.Time_idTimeCasa = t.idTime THEN dp.golsCasa ELSE dp.golsVisitante END), 0) AS GolsMarcados, "
            + "COALESCE(SUM(CASE WHEN dp.Time_idTimeCasa = t.idTime THEN dp.golsVisitante ELSE dp.golsCasa END), 0) AS GolsSofridos "
            + "FROM Time t "
            + "LEFT JOIN Detalhes_Partida dp ON dp.Time_idTimeCasa = t.idTime OR dp.Time_idTimeVisitante = t.idTime "
            + "LEFT JOIN Partida p ON dp.Partida_idPartida = p.idPartida "
            + "LEFT JOIN Rodada r ON p.Rodada_idRodada = r.idRodada "
            + "LEFT JOIN Campeonato c ON r.Campeonato_idCampeonato = c.idCampeonato " + "WHERE c.nome = ? "
            + "GROUP BY t.idTime, t.nome " + "ORDER BY Pontos DESC, GolsMarcados DESC";

    List<Classificacao> classificacao = new ArrayList();
```

Cadastro Jogador

```java
public void inserirPessoaJogador(int idPessoa, String nome, String dataNascimento, String cpf) {

    String sql = "INSERT INTO Pessoa " + "(idPessoa, nome, data_nascimento, cpf) " + "VALUES(?, ?, ?, ?)";

    try {

        Connection conn = ConnectionFactory.getConnection();
        java.sql.PreparedStatement pstm = conn.prepareStatement(sql);

        // Definindo os valores dos parâmetros
        pstm.setInt(1, idPessoa); // id
        pstm.setString(2, nome); // nome
        pstm.setString(3, dataNascimento); // data de nascimento
        pstm.setString(4, cpf); // idPessoa

        pstm.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
    }

}
```

Atualização Jogador

```java
public static void atualizarJogador(int idPessoa, String name, String dataNascimento) {

    String sql = "update Pessoa " + "set nome = ?, data_nascimento = ? " + "where idPessoa = ?";

    try {

        Connection conn = ConnectionFactory.getConnection();
        java.sql.PreparedStatement pstm = conn.prepareStatement(sql);

        // Definindo os valores dos parâmetros
        pstm.setString(1, name); // nome
        pstm.setString(2, dataNascimento); //data de nascimento
        pstm.setInt(3, idPessoa); // idPessoa

        pstm.executeUpdate();

        System.out.println("Dados atualizados com sucesso!");
        System.out.println("");

    } catch (SQLException e) {
        e.printStackTrace();
    }

}
```