INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

JOÃO PEDRO VIANA RODRIGUES

SISTEMA DE GERENCIAMENTO DE TREINOS

CAMPOS DO JORDÃO
2024

SUMÁRIO

1 INTRODUÇÃO

Este sistema oferece a implementação sofisticada e pratica que otimiza a passagem de treinos, dieta e metas entre educador fisico e atleta. O sistema permite que o aluno receba atualizações sobre tudo relacionado a seus treinos, desde dicas para melhores execuções até qual será sua nova rotina de treinos durante a semana, todos atualizados por seu professor, sendo específicos para cada pessoa, notando suas particularidades, dificuldades e metas.
O sistema conseguirá analisar e gerar sugestões para o próprio professor com base no seu banco de dados gerados pelos seus proprios algoritmos, podendo assim, auxiliar no desempenho e criação de treinos passados para cada atleta.

Palavras-Chave: Aplicativo de treino, Academia, Personal, Atleta.

2 OBJETIVO

Este trabalho tem como objetivo facilitar o desenvolvimento de atletas e sua conversa com seus treinadores, deixando assim, ambas as partes conscientes de todo o processo, podendo-se assim ter maior facilidade de alcançar suas metas e moldar seus resultados.

3 METODOLOGIA

3.1 Considerações Iniciais

A etapa inicial do projeto envolve ter o conhecimento das necessidades e demandas que um treinador tem para entender as necessidades de seus alunos a fim de montar seus treinos com base nas suas dificuldades e metas.

3.2 Coleta de Requisitos

A coleta de requisitos será realizada conversando com profissionais da área, entendendo suas demandas e necessidades com base em suas experiências.

3.3 Projeto de Dados:

O projeto de dados será elaborado para organizar e estruturar as informações que seriam armazenadas no sistema. Identificando os atributos relevantes de cada entidade e definindo as relações entre elas, garantindo assim a integridade e a consistência dos dados armazenados.

4 O que o Sistema Terá:

- Um dashboard interativo com o usuário, onde se encontram seus treinos da semana, últimas modificações, seus resultados, suas metas a longo prazo e com datas definidas.

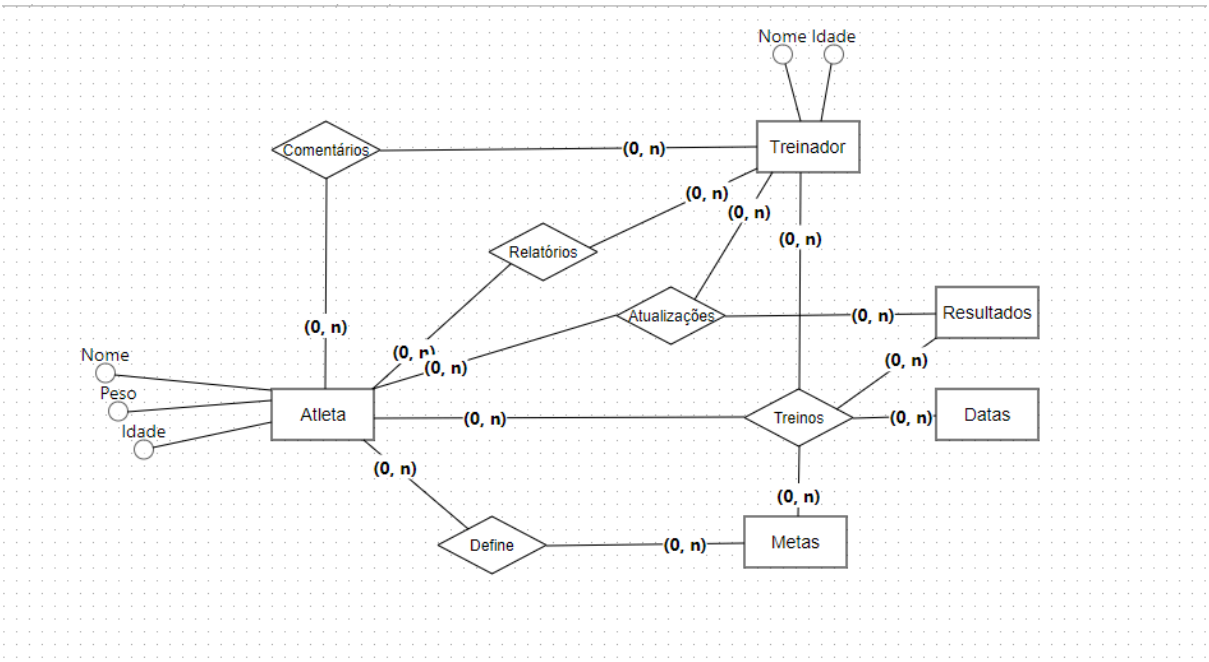
- O atleta poderá marcar seus treinos como feitos e adicionar comentários para que seu professor possa ver em sua interface, facilitando assim a melhoria de sua rotina em prol de seu resultado.

- O professor poderá atualizar seus treinos, definir suas metas de acordo com consultas e relatórios de seus alunos e atualizar os resultados de seus alunos na plataforma.

5 Implementação do Banco de Dados:

Utilizei o BrModelo para a implementação do banco de dados, gerando o modelo conceitual do sistema.

5.1 Modelo Conceitual



5.2 Regras de Negócio:

- Um usuário (atleta) deve ter um nome, nome de usuário, telefone e e-mail únicos.
- O nome de usuário e e-mail não podem ser alterados uma vez que criados.
- O tipo de usuário deve ser definido no momento da criação do perfil do usuário.
- O tipo de usuário pode ser "Treinador" ou "Atleta".
- Treinadores devem fornecer credenciais de qualificação adequadas.
- Os treinadores devem registrar o progresso dos usuários em termos de metas e objetivos individuais.
- Os usuários devem ter acesso a relatórios que mostram seu progresso ao longo do tempo.
- Os treinadores devem adaptar os treinos com base nas necessidades e capacidades individuais dos usuários.
- Deve haver uma revisão para ajustar os planos de treino conforme necessário.
- Deve haver uma plataforma para comunicação direta entre usuários e treinadores para discutir objetivos, planos de treino e feedback.
- Oferecer suporte ao cliente para resolver problemas técnicos, questões de agendamento ou reclamações relacionadas ao serviço.

6 Modelo Físico:

Para a criação do modelo físico utilizamos os seguintes scripts:

Para a criação e uso do banco:

```
CREATE DATABASE APP;
USE APP;
```

Criação da tabela Atleta:

```sql
CREATE TABLE Atleta (
    idAtleta INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    idade INT NOT NULL,
    peso DECIMAL(5,2) NOT NULL
);
```

Criação da tabela Treinador:

```sql
CREATE TABLE Treinador (
    idTreinador INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    idade INT NOT NULL
);
```

Criação da tabela Metas:

```sql
CREATE TABLE Metas (
    idMetas INT AUTO_INCREMENT PRIMARY KEY,
    descricao VARCHAR(255) NOT NULL
);
```

Criação da tabela Resultados:

```sql
CREATE TABLE Resultados (
    idResultados INT AUTO_INCREMENT PRIMARY KEY,
    descricao VARCHAR(255) NOT NULL
);
```

Criação da tabela Datas:

```sql
CREATE TABLE Datas (
    idDatas INT AUTO_INCREMENT PRIMARY KEY,
    data DATE NOT NULL
);
```

Criação da tabela Comentarios:

```sql
CREATE TABLE Comentarios (
    idComentario INT AUTO_INCREMENT PRIMARY KEY,
    idTreinador INT NOT NULL,
    idAtleta INT NOT NULL,
    comentario TEXT NOT NULL,
    FOREIGN KEY (idTreinador) REFERENCES Treinador(idTreinador),
    FOREIGN KEY (idAtleta) REFERENCES Atleta(idAtleta)
);
```

Criação da tabela Treinos:

```sql
CREATE TABLE Treinos (
    idTreino INT AUTO_INCREMENT PRIMARY KEY,
    idResultados INT NOT NULL,
    idDatas INT NOT NULL,
    idMetas INT NOT NULL,
    idAtleta INT NOT NULL,
    idTreinador INT NOT NULL,
    FOREIGN KEY (idResultados) REFERENCES Resultados(idResultados),
    FOREIGN KEY (idDatas) REFERENCES Datas(idDatas),
    FOREIGN KEY (idMetas) REFERENCES Metas(idMetas),
    FOREIGN KEY (idAtleta) REFERENCES Atleta(idAtleta),
    FOREIGN KEY (idTreinador) REFERENCES Treinador(idTreinador)
);
```

Criação da tabela Relatórios:

```sql
CREATE TABLE Relatorios (
    idRelatorio INT AUTO_INCREMENT PRIMARY KEY,
    idAtleta INT NOT NULL,
    idTreinador INT NOT NULL,
    descricao TEXT,
    FOREIGN KEY (idAtleta) REFERENCES Atleta(idAtleta),
    FOREIGN KEY (idTreinador) REFERENCES Treinador(idTreinador)
);
```

Criação da tabela Atualizações:

```sql
CREATE TABLE Atualizacoes (
    idAtualizacao INT AUTO_INCREMENT PRIMARY KEY,
    idAtleta INT NOT NULL,
    idResultados INT NOT NULL,
    idTreinador INT NOT NULL,
    descricao TEXT,
    FOREIGN KEY (idAtleta) REFERENCES Atleta(idAtleta),
    FOREIGN KEY (idResultados) REFERENCES Resultados(idResultados),
    FOREIGN KEY (idTreinador) REFERENCES Treinador(idTreinador)
);
```

7 Consultas:

1. Listar todos os atletas com mais de 30 anos:

```sql
SELECT nome, idade
FROM Atleta
WHERE idade > 30;
```

2. Encontrar o atleta mais pesado

```sql
SELECT nome, peso
FROM Atleta
ORDER BY peso DESC
LIMIT 1;
```

3. Calcular a média de idade dos atletas

```sql
SELECT AVG(idade) AS media_idade
FROM Atleta;
```

4. Exibir os treinadores que comentaram sobre atletas

```sql
SELECT DISTINCT Treinador.nome
FROM Treinador
JOIN Comentarios ON Treinador.idTreinador = Comentarios.idTreinador;
```

5. Listar atletas com metas associadas

```sql
SELECT Atleta.nome, Metas.descricao
FROM Atleta
JOIN Treinos ON Atleta.idAtleta = Treinos.idAtleta
JOIN Metas ON Treinos.idMetas = Metas.idMetas;
```

6. Contar o total de resultados registrados

```sql
SELECT COUNT(*) AS total_resultados
FROM Resultados;
```

7. Exibir os treinadores que possuem mais de 5 treinos cadastrados

```sql
SELECT Treinador.nome, COUNT(*) AS total_treinos
FROM Treinador
JOIN Treinos ON Treinador.idTreinador = Treinos.idTreinador
GROUP BY Treinador.idTreinador
HAVING total_treinos > 5;
```

8. Encontrar atletas que nunca receberam comentários

```sql
SELECT nome
FROM Atleta
WHERE idAtleta NOT IN (
    SELECT idAtleta
    FROM Comentarios
);
```

9. Listar todos os treinos ocorridos em uma data específica

```sql
SELECT Treinos.idTreino, Datas.data
FROM Treinos
JOIN Datas ON Treinos.idDatas = Datas.idDatas
WHERE Datas.data = '2024-01-01';
```

10. Calcular o total de metas por treinador

```sql
SELECT Treinador.nome, COUNT(DISTINCT Treinos.idMetas) AS total_metas
FROM Treinador
JOIN Treinos ON Treinador.idTreinador = Treinos.idTreinador
GROUP BY Treinador.idTreinador;
```

11. Listar atletas com o maior número de treinos

```sql
SELECT Atleta.nome, COUNT(*) AS total_treinos
FROM Atleta
JOIN Treinos ON Atleta.idAtleta = Treinos.idAtleta
GROUP BY Atleta.idAtleta
ORDER BY total_treinos DESC
LIMIT 1;
```

12. Exibir as datas dos treinos de um atleta específico

```sql
SELECT Datas.data
FROM Datas
JOIN Treinos ON Datas.idDatas = Treinos.idDatas
WHERE Treinos.idAtleta = 1;
```

13. Contar o total de atualizações feitas por treinador

```sql
SELECT Treinador.nome, COUNT(*) AS total_atualizacoes
FROM Treinador
JOIN Atualizacoes ON Treinador.idTreinador = Atualizacoes.idTreinador
GROUP BY Treinador.idTreinador;
```

14. Exibir a descrição dos resultados de um atleta

```sql
SELECT Resultados.descricao
FROM Resultados
JOIN Treinos ON Resultados.idResultados = Treinos.idResultados
WHERE Treinos.idAtleta = 2;
```

15. Identificar o treinador que comentou mais

```sql
SELECT Treinador.nome, COUNT(*) AS total_comentarios
FROM Treinador
JOIN Comentarios ON Treinador.idTreinador = Comentarios.idTreinador
GROUP BY Treinador.idTreinador
ORDER BY total_comentarios DESC
LIMIT 1;
```

16. Exibir a maior diferença de peso entre atletas

```sql
SELECT MAX(peso) - MIN(peso) AS diferenca_peso
FROM Atleta;
```

17. Listar os 5 atletas mais jovens

```sql
SELECT nome, idade
FROM Atleta
ORDER BY idade ASC
LIMIT 5;
```

18. Encontrar treinadores que treinaram mais de 3 atletas

```sql
SELECT Treinador.nome, COUNT(DISTINCT Treinos.idAtleta) AS total_atletas
FROM Treinador
JOIN Treinos ON Treinador.idTreinador = Treinos.idTreinador
GROUP BY Treinador.idTreinador
HAVING total_atletas > 3;
```

19. Exibir os treinos realizados em um intervalo de datas

```sql
SELECT Treinos.idTreino, Datas.data
FROM Treinos
JOIN Datas ON Treinos.idDatas = Datas.idDatas
WHERE Datas.data BETWEEN '2024-01-01' AND '2024-12-31';
```

20. Calcular a média de peso dos atletas que possuem mais de 5 treinos

```sql
SELECT AVG(Atleta.peso) AS media_peso
FROM Atleta
JOIN Treinos ON Atleta.idAtleta = Treinos.idAtleta
GROUP BY Atleta.idAtleta
HAVING COUNT(*) > 5;
```

21. Exibir o último treino registrado

```sql
SELECT Treinos.idTreino, Datas.data
FROM Treinos
JOIN Datas ON Treinos.idDatas = Datas.idDatas
ORDER BY Datas.data DESC
LIMIT 1;
```

22. Encontrar a meta mais comum entre os treinos

```sql
SELECT Metas.descricao, COUNT(*) AS total
FROM Metas
JOIN Treinos ON Metas.idMetas = Treinos.idMetas
GROUP BY Metas.idMetas
ORDER BY total DESC
LIMIT 1;
```

23. Exibir o total de treinos realizados por data

```sql
SELECT Datas.data, COUNT(*) AS total_treinos
FROM Datas
JOIN Treinos ON Datas.idDatas = Treinos.idDatas
GROUP BY Datas.data;
```

24. Encontrar atletas que possuem atualizações associadas a resultados específicos

```sql
SELECT Atleta.nome, Resultados.descricao
FROM Atleta
JOIN Atualizacoes ON Atleta.idAtleta = Atualizacoes.idAtleta
JOIN Resultados ON Atualizacoes.idResultados = Resultados.idResultados;
```

25. Exibir os relatórios feitos para um atleta específico

```sql
SELECT Relatorios.descricao
FROM Relatorios
WHERE Relatorios.idAtleta = 3;
```

26. Contar o número de comentários feitos por treinador

```sql
SELECT Treinador.nome, COUNT(*) AS total_comentarios
FROM Treinador
JOIN Comentarios ON Treinador.idTreinador = Comentarios.idTreinador
GROUP BY Treinador.idTreinador;
```

27. Listar atletas sem treinos registrados

```sql
SELECT nome
FROM Atleta
WHERE idAtleta NOT IN (
    SELECT idAtleta
    FROM Treinos
);
```

28. Identificar datas com mais de 10 treinos

```sql
SELECT Datas.data, COUNT(*) AS total_treinos
FROM Datas
JOIN Treinos ON Datas.idDatas = Treinos.idDatas
GROUP BY Datas.data
HAVING total_treinos > 10;
```

29. Calcular a soma total dos pesos dos atletas

```sql
SELECT SUM(peso) AS soma_peso
FROM Atleta;
```

30. Encontrar os 3 atletas mais comentados

```sql
SELECT Atleta.nome, COUNT(*) AS total_comentarios
FROM Atleta
JOIN Comentarios ON Atleta.idAtleta = Comentarios.idAtleta
GROUP BY Atleta.idAtleta
ORDER BY total_comentarios DESC
LIMIT 3;
```