Instituto Federal de Educação, Ciência e Tecnologia do Estado de São Paulo

Banco de dados

Camile Lopes Franco de Macedo

Gabriel de Souza Reis

Pedro Machado Marchi

Projeto Interdisciplinar (NutriLife)

São Paulo, 2023

Notação Textual – Projeto banco de dados (NutriLife)

USUARIO (cpf, pesoAtual, pesoInicial, altura, idade, nome, restricoes)

PROFISSIONAL (idProfissional, certiificado, nome, cpf, endereco, dataNacimento)

HISTORICO (idHistorico, cpfUsuario, idDieta)

cpfUsuario REFERENCIA USUARIO(cpf)

IdDieta REFERENCIA DIETA(idDieta)

DIETA (idDieta, dataCriacao, periodo, tipo, idProfissional, cpfUsuario)

cpfUsuario REFERENCIA USUARIO(cpf)

IdProfissional REFERENCIA PROFISSIONAL(idPofissional)

ATIVIDADES (idAtividades, nome, instruções, nivelDificuldade, feedback)

REFEICOES (idRefeicao, descricao, feedback)

DIETAS_POSSUI_ATIVIDADES(idDieta, idAtividades)

IdDieta REFERENCIA DIETA(idDieta)

IdAtividade REFERENCIA ATIVIDADES(idAtividades)

DIETAS_POSSUI_REFEIÇÕES(idDieta, idRefeicao)

IdDieta REFERENCIA DIETA(idDieta)

IdRefeicao REFERENCIA REFEICOES(idRefeicao)

Universo de Discurso

O sistema “NutriLife” é uma empresa que cuida principalmente do controle de dietas e exercícios. Muitas pessoas com a vida acelerada e sem flexibilidade de tempo, acarretando em uma falta de alimentação saudável e sem conhecimento de exercícios que podem auxiliar para um bem-estar maior.

Para se beneficiar com o NutriLife, o usuário precisa realizar o login com seus dados como cpf, nome, restrições e preferências alimentares, peso atual e altura, e senha para o aplicativo. Depois de criado, o usuário deve fazer seu login e acessar o aplicativo, onde é possível realizar consulta com profissionais, sendo esses também registrados na plataforma com alguns dados como nome, cpf, endereço, data de nascimento e certificado.

Além da interação usuário profissional no qual o usuário terá uma dieta personalizada e exercícios recomendados pelos profissionais, o sistema deverá guardar no histórico de cada usuário suas dietas e informações como sua data de criação, o período em que se é recomendado praticá-la e o tipo dela (se é pra emagrecimento, vegana, low carb etc…) e o profissional que a criou.

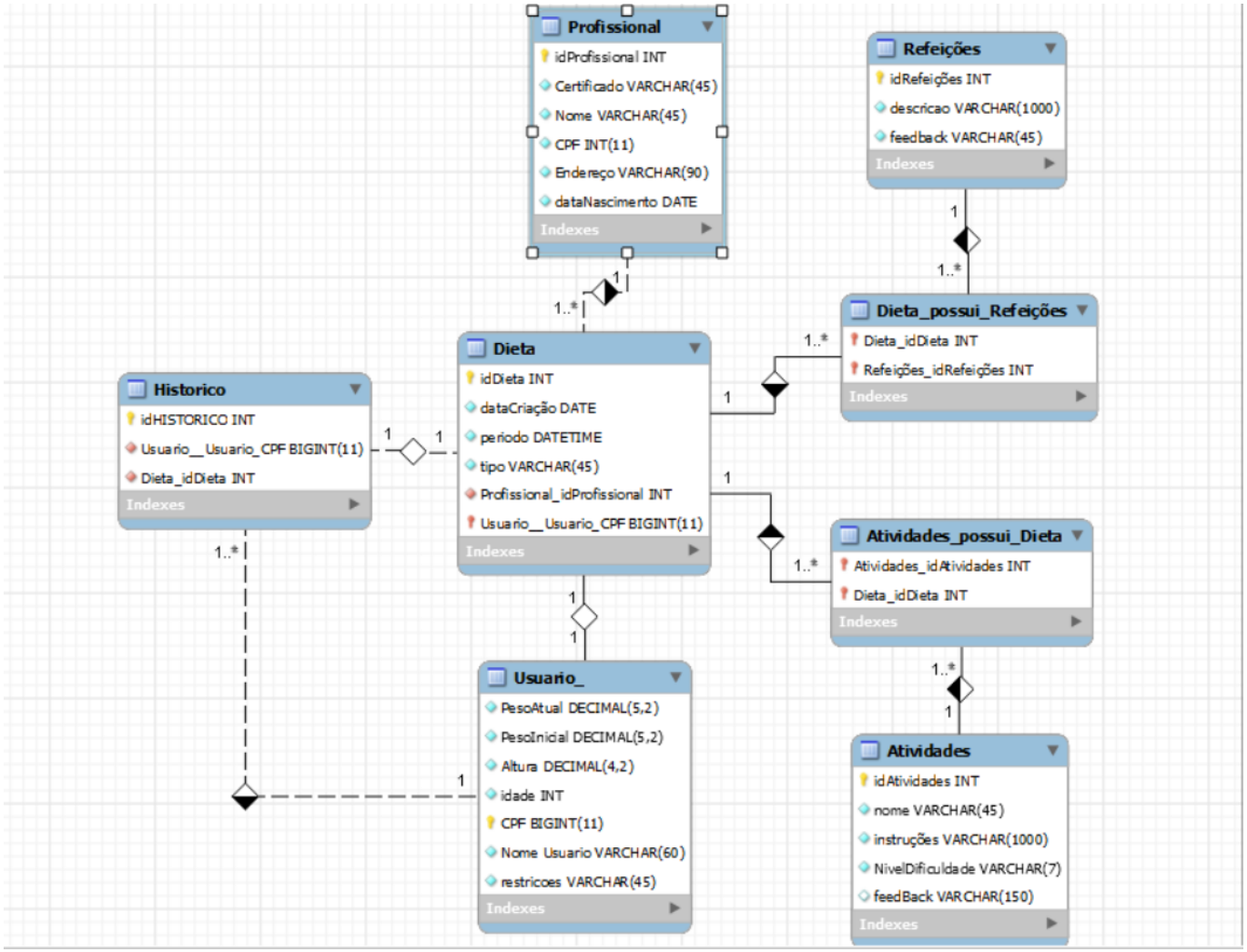
As atividades que só podem ser criadas por profissionais devem conter nome, instrução e seu nível de dificuldade.

As refeições montadas somente por profissionais devem conter descrição com suas informações.

O usuário deve ter a capacidade de avaliar profissionais, em que essas avaliações/feedbacks serão armazenados pelo históricos.

Os profissionais podem criar várias dietas, e um usuário pode ter mais de uma dieta, refeições e exercícios que estão todas ligadas entre si.

DER – Diagrama Entidade Relacionamento



Script

```
#DROP DATABASE nutriLife;
CREATE DATABASE IF NOT EXISTS nutriLife;
USE nutriLife;

# Criação da tabela USUARIO
CREATE TABLE IF NOT EXISTS usuario (
cpf bigint(11) not null,
pesoAtual decimal(5,2) not null,
pesoInicial decimal(5,2) not null,
altura decimal(4,2) not null,
idade int not null,
nome varchar(45) not null,
restricoes varchar(200),

primary key(cpf)
);

INSERT INTO usuario (cpf, pesoAtual, pesoInicial, altura, idade, nome, restricoes)
VALUES (12345678910, 75.50, 80.00, 1.75, 30, 'João Silva', 'Glúten'),
    (98765432109, 65.20, 70.00, 1.60, 28, 'Maria Santos', 'Lactose'),
    (45678912345, 85.00, 90.00, 1.80, 35, 'Carlos Oliveira', null),
    (11122233344, 70.80, 75.00, 1.68, 32, 'Ana Pereira', null),
    (55566677788, 62.40, 65.00, 1.55, 27, 'Pedro Santos', 'Ovo'),
    (44746647768,59.00, 59.00, 1.79, 20, 'Gabriel Dias', null);
```

```
#Criação da tabela profissional
CREATE TABLE IF NOT EXISTS profissional (
idProfissional int auto_increment not null,
certificado varchar(135) not null,
nome varchar(45) not null,
cpf bigint(11) not null,
endereco varchar(135) not null,
dataNascimento date,

primary key(idProfissional)
);

#insert into profissional
#values(),
INSERT INTO profissional (certificado, nome, cpf, endereco, dataNascimento)
VALUES ('Certificado NutriVida', 'Kaio Santos', 12345678910, 'Rua A, 123', '1990-05-15'),
    ('Certificado USP', 'Mariana Jesus', 98765432109, 'Rua B, 456', '1985-10-20'),
    ('Certificado CursoOnline', 'Carlos Gabriel', 45678912345, 'Rua C, 789', '1995-07-08'),
    ('Certificado Fatec', 'Fabio Augusto', 11122233344, 'Rua D, 789', '1992-09-12'),
    ('Certificado NutriVida', 'Jaqueline Dias', 55566677788, 'Rua E, 321', '1998-03-25');
```

```
#Criação da tabela dieta
CREATE TABLE IF NOT EXISTS dieta (
idDieta int auto_increment not null,
dataCriacao date not null,
periodo datetime not null,
tipo varchar(45),
idProfissional int not null,
cpf bigint(11) not null,

primary key(idDieta),
foreign key(idProfissional) references profissional(idProfissional),
foreign key(cpf) references usuario(cpf)
);

INSERT INTO dieta (dataCriacao, periodo, tipo, idProfissional, cpf)
VALUES ('2022-01-01', '2022-02-01 00:00:00', 'Vegana', 1, 12345678910),
      ('2022-02-01', '2022-05-01 00:00:00', 'Emagrecimento', 2, 98765432109),
      ('2022-03-01', '2022-07-01 00:00:00', 'LowCarb', 3, 45678912345),
     ('2022-04-01', '2022-09-01 00:00:00', 'LowCarb', 4, 11122233344),
      ('2022-05-01', '2022-10-01 00:00:00', 'Vegana', 5, 55566677788);
```

```
#Criação da tabela historico
CREATE TABLE IF NOT EXISTS historico (
idHistorico int auto_increment not null,
idDieta int not null,
cpf bigint(11) not null,

primary key(idHistorico),
foreign key(idDieta) references dieta(idDieta),
foreign key(cpf) references usuario(cpf)
);

INSERT INTO historico (idHistorico, idDieta, cpf)
VALUES (1, 1, 12345678910),
      (2, 2, 98765432109),
      (3, 3, 45678912345),
      (4, 4, 11122233344);

#Criação da tabela atividades
CREATE TABLE IF NOT EXISTS atividades (
idAtividade int auto_increment not null,
nome varchar(45) not null,
instrucoes varchar(1000) not null,
nivelDificuldade varchar(45) not null,
feedBack varchar(135),
idDieta int not null,
 primary key(idAtividade),
foreign key(idDieta) references dieta(idDieta)
);
```

```
INSERT INTO atividades (nome, instrucoes, nivelDificuldade, feedBack, idDieta)
VALUES ('Caminhada', 'Realize uma caminhada de 30 minutos.', 'Fácil', 'Ótimo!', 1),
    ('Agachamentos', 'Realize 3 séries de 10 agachamentos.', 'Moderado', 'Me senti mais forte!', 1),
    ('Flexões', 'Realize 3 séries de 12 flexões.', 'Moderado', 'Consegui completar todas!', 2),
    ('Alongamentos', 'Realize uma sequência de alongamentos por 10 minutos.', 'Fácil', 'Me senti mais mais flexível!', 3),
    ('Corrida', 'Corra por 5 km a um ritmo moderado.', 'Difícil', 'Melhorei meu tempo!', 4);

#Criação da tabela refeições
CREATE TABLE IF NOT EXISTS refeicoes (
idRefeicao int auto_increment not null,
descricao varchar(1000) not null,
feedback varchar(135),
idDieta int not null,

primary key(idRefeicao),
foreign key(idDieta) references dieta(idDieta)
);

INSERT INTO refeicoes (descricao, feedback, idDieta)
VALUES ('Café da manhã: Omelete de claras com vegetais e uma fatia de pão integral.', 'Delicioso!', 1),
    ('Almoço: Filé de frango grelhado com arroz integral e salada de folhas.', 'Saboroso e saudável!', 1),
    ('Lanche da tarde: Smoothie de frutas vermelhas com iogurte natural.', 'Refrescante e nutritivo!', 2),
    ('Jantar: Salmão assado com legumes ao vapor.', 'Uma refeição equilibrada!', 3),
    ('Ceia: Chá de camomila com biscoitos integrais.', 'Relaxante antes de dormir!', 4);
```

#-- QUERYS --#

Q1 - Exibir usuarios que possuem uma restrição alimentar.

```
SELECT nome, idade FROM usuario WHERE restricoes IS NOT NULL;
```

Q2 - Exibir todos os usuários com idade superior a 25 anos e altura maior que 1,70m.

```
SELECT * FROM usuario WHERE idade > 25 AND altura > 1.70;
```

Q3 - Exibir o nome e o peso atual dos usuários em ordem decrescente de peso atual.

```
SELECT nome, pesoAtual FROM usuario ORDER BY pesoAtual DESC;
```

Q4 - Exibir o nome do profissional e a quantidade de usuários que ele atende.

```
SELECT p.nome, COUNT(u.cpf) AS quantidade_usuarios
FROM profissional p
INNER JOIN dieta d ON p.idProfissional = d.idProfissional
INNER JOIN usuario u ON d.cpf = u.cpf
GROUP BY p.idProfissional;
```

Q5 - Média de idade dos usuários.

```
SELECT AVG(idade) AS media_idade FROM usuario;
```

Q6 - Exibir a quantidade de dietas de cada tipo.

```
SELECT tipo, COUNT(*) AS quantidade_dietas
FROM dieta
GROUP BY tipo;
```

```
# Q7 - Exibir a data de criação da dieta e a quantidade de atividades associadas a ela:
SELECT d.dataCriacao, COUNT(*) AS quantidade_atividades
FROM dieta d
JOIN atividades a ON d.idDieta = a.idDieta
GROUP BY d.idDieta;

# Q8 - Descrição das refeições e o feedback associado a elas.
SELECT descricao, feedback FROM refeicoes;

# Q9 - Dietas que possuem atividades de nível difícil.
SELECT d.*
FROM dieta d
INNER JOIN atividades a ON d.idDieta = a.idDieta
WHERE a.nivelDificuldade = 'Difícil';

# Q10 - Nome dos usuários e a quantidade de dietas que eles já seguiram.
SELECT u.nome, COUNT(h.idDieta) AS quantidade_dietas
FROM usuario u
LEFT JOIN historico h ON u.cpf = h.cpf
GROUP BY u.cpf;

#Q11 - Nome dos profissionais que possuem certificados emitidos pela "Certificado NutriVida".
SELECT nome
FROM profissional
WHERE certificado = 'Certificado NutriVida';
```

Q12 - Quantidade de atividades de cada nível de dificuldade.

```
SELECT nivelDificuldade, COUNT(*) AS quantidade_atividades
FROM atividades
GROUP BY nivelDificuldade;
```