UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE CIÊNCIA APLICADAS E EDUCAÇÃO, CCAE

DEPARTAMENTO DE CIÊNCIAS EXATAS, DCX

SEGUNDA AVALIAÇÃO DA DISCIPLINA DE BANCO DE DADOS

Projeto CanDoIt

Rio Tinto

Novembro de 2020

Breno Oliveira Queiroz, 20160121960
Cleyson Henrique Oliveira Bezerra, 20180104297
Francisco Artur Bernardo Neto, 20180024759
Gabriel Gurjão Camargo Campos, 20180023199
Gabriel Ribeiro Dias, 20180032071

SEGUNDA AVALIAÇÃO DA DISCIPLINA DE BANCO DE DADOS

Projeto CanDoIt

Projeto da disciplina de Banco de dados para a nota da segunda avaliação.

Professor: Leandro Figueiredo Alves

Rio Tinto

Novembro de 2020

Resumo

A ideia geral do aplicativo gira em torno dos usuários criarem metas que desejam cumprir, que podem ser: superação de vícios, atingir uma determinada quantia em dinheiro, gerir de forma eficiente as suas tarefas e criar uma rotina diária de exercícios. Assim motivamos os usuários a se sentirem cada vez mais incentivados a realizar suas conquistas pessoais.

1. Introdução

Atualmente as pessoas tem dificuldade de manter o foco e a motivação para alcançar seus objetivos e metas de vida, seja por falta de tempo em consequência de uma vida muito atarefada em que não conseguem organizar sua vida pessoal, e por falta de foco não conseguem manter-se motivadas em consequência da quantidade de tarefas em que precisa se concentrar no seu dia a dia e por isso acaba se esquecendo dos seus objetivos. Existem também pessoas que sofrem com vícios e buscam uma maneira para que consigam se livrar deles, porém não falham ao tentar fazer isso sozinhos por falta de motivação. Pessoas que querem ter uma rotina saudável de exercícios porém não conseguem. E dentro deste contexto, surgiram os apps coach, de gerenciamento de tarefas, de gerenciamento de rotina, que buscam ajudar as pessoas a superar essas dificuldades.

Pensando nisso, idealizamos um projeto que consiste em um aplicativo para android, que tem por objetivo ajudar as pessoas a cumprimerem seus objetivos e metas através do incentivo visual dinâmico em uma interface intuitiva. O aplicativo iria dispor em tela as metas criadas pelo usuário e seu progresso em relação ao cumprimento das metas, junto com frases motivacionais para cada tipo de meta. As metas são organizadas em categorias.

Para cada categoria de meta e objetivo, o modo como o usuário cria essa meta vai mudar, e a forma como ele vai avançar no cumprimento da meta também. Além do usuário criar suas metas através das categorias pré-estabelecidas no aplicativo, ele terá a opção de criar uma meta personalizada, onde ele irá escolher a forma como quer evoluir na meta, bem como as tarefas que ele deseja cumprir.

Os usuários não precisam estar logados para criarem metas. Cada meta concluída será salva e mostrada em uma lista de metas. Para cada meta concluída, o usuário receberá uma conquista, como uma premiação por ter concluída a meta. As premiações poderão ser visualizadas pelo usuário em uma lista de conquistas. Além da criação, o usuário também poderá remover metas que decidiu não mais concluir.

2. Requisitos

2.1 Estórias de usuário

#1 - Criar metas de vícios padrão

Eu **como usuário** Quero **criar uma meta para vícios** Para **eu superar meu vício**

#2 - Criar meta financeira padrão

Eu **como usuário** Quero **criar uma meta financeira** Para **economizar e me ajustar a me manter focado na minha meta**

#3 - Criar meta de rotina de exercícios padrão

Eu **como usuário** Quero **criar uma rotina de exercício** Para me **auxiliar nos treinos**

#4 - Criar meta de dieta padrão

Eu **como usuário** Quero **criar uma meta para dieta** Para **me manter focado na dieta**

#5 - Reiniciar contador de tempo da meta de vício

Eu **como usuário** Quero **poder reiniciar o contador do tempo que estou livre do vício** Para **poder recomeçar minha meta**

#6 - Adicionar valor na meta ao contador de progresso da meta financeira

Eu **como usuário** Quero **poder adicionar quanto eu consegui juntar dinheiro** Para **poder acompanhar meu progresso atual**

#7 - Subtrair valor do contador de progresso da meta financeira

Eu **como usuário**
Quero **poder subtrair da quantia que havia juntado**
Para **poder acompanhar meu progresso atual**

#8 - Botão de cancelar Meta

Eu **como usuário**
Quero **poder cancelar a meta**
Para **poder ajustar ou iniciar outra meta**

#9 - Modificar a meta personalizada

Eu **como usuário**
Quero **que minhas metas possam ser modificadas**
Para **que eu consiga acrescentar ou remover tarefas para melhor evolução na meta**

#10 - Premiação por conclusão de meta

Eu **como programador**
Quero **que o usuário premiado por cumprir uma meta**
Para **que ele se sinta satisfeito em ter conseguido concluir a meta**

#11 - Frases Motivacionais

Eu **como programador**
Quero **o usuário receba notificações**
Para **se manter focado e motivado a cumprir suas metas**

#12 - Barra de progresso

Eu **como programador**
Quero **que as metas tenham uma barra de progresso**
Para **que o usuário consiga acompanhar a sua evolução na meta.**

#13 - Lista de metas concluídas

Eu **como programador**
Quero **que as metas que o usuário já concluiu sejam salvas em uma lista**
Para **que ele possa saber quais das metas ele já concluiu**

#14 - Filtrar metas pelo tipo

Eu **como usuário** Quero **filtrar as minhas metas criadas** Para **que eu possa buscar uma uma meta com maior facilidade**

#15 - Botão de encerrar meta concluída

Eu **como usuário** Quero **encerrar a meta que eu criei** Para **que consiga encerrar as metas que eu já acho que concluí**

#16 - Botão de dobrar a meta

Eu **como usuário** Quero **dobrar as minha meta** Para **que eu possa estender a minha meta e continue evoluindo mais nela**

2.2 Casos de Uso

1 Caso de Uso

1.1 Identificação do caso de uso

UC_01 - Criar meta de vício

1.2 Descrição do caso de uso

Este caso de uso está relacionado à criação de uma meta de vício. Portanto, por meio deste caso de uso, o usuário pode criar uma meta do tipo vício. Uma meta de vício consiste em uma meta relacionada aos vícios que o usuário enfrente e gostaria de ter algum incentivo para livrar-se dele, desde o uso excessivo das redes sociais, até uso de cigarro e bebidas.

1.3 Atores envolvidos

1. Usuário

1.4 Pré-condições

1. O usuário deve estar com o aplicativo aberto, na tela principal.

1.5 Fluxos de Eventos

1.5.1 FB - Fluxo Básico

1. O usuário solicita a criação de uma meta;
2. O usuário informa o vício que ele deseja livrar-se com a ajuda do app;(*FE_01 - Obrigatoriedade da escolha de um vício*)
3. O usuário informa as tarefas que ele deseja listar na sua meta;(*FE_02 - Obrigatoriedade de um nome para a tarefa*)
4. O usuário cria a meta
5. O sistema lista a meta criada na lista de metas criadas da tela principal.
6. Caso de uso é encerrado.

1.5.2 Fluxos Alternativos

1.5.3 Fluxos de Exceção

FE_01 - Obrigatoriedade da escolha de um vício

1. Este fluxo se inicia quando o usuário tenta prosseguir na criação da meta de vicio sem ter setado o vício que deseja se livrar.
2. O sistema deve mostrar uma mensagem informando o usuário que ele deve escolher um vício para poder continuar (*MSG_01 - objetivo não escolhido*);
3. O sistema retorna para o **FB_02;**

FE_02 - Obrigatoriedade de um nome para a tarefa

1. Este fluxo se inicia quando o usuário tenta prosseguir na criação de um meta com uma tarefa criada em branco;
2. O sistema deve mostrar uma mensagem informando o usuário acerca da meta em branco(*MSG_01 - Meta em branco*);
3. O sistema deve retornar para o **FB_03;**

1.6 Pós-condições

1. O sistema deve adicionar a meta a uma lista de metas criadas na tela principal do app.

2. O sistema computa os dados atuais de progresso do usuário na meta.

2 Caso de Uso

2.1 Identificação do caso de uso

UC_02 - Criar meta de dieta

2.2 Descrição do caso de uso

Este caso de uso está relacionado a criação da meta de dieta , o usuário pode criar uma meta e tarefas para auxiliar em seu objetivo.Essa meta estará dividida em 2 tipos “ganhar peso” e “perder peso” .

2.3 Atores envolvidos

1. usuário

2.4 Pré-condições

1 . usuário precisa estar na tela principal do aplicativo

2.5 Fluxos de Eventos

2.5.1 FB - Fluxo Básico

1. O usuário solicita a criação de uma meta;
2. O usuário seleciona a criação de meta de dieta;
3. O usuário seleciona o objetivo de perder peso(*FA_1 - Usuário seleciona o objetivo de ganhar peso*)
4. O usuário clica em adicionar tarefas
5. O usuário preenche os dados da tarefa:

- Nome da tarefa;

6. O usuário clica em “criar meta”
7. Sistema adiciona a meta da lista de metas criadas da tela principal;
8. Caso de meta encerrado;

2.5.2 Fluxos Alternativos

2.5.3 Fluxos de Exceção

FE_01 - Obrigatoriedade da escolha do objetivo

4. Este fluxo se inicia quando o usuário tenta prosseguir na criação da meta de dieta sem ter setado o objetivo .
5. O sistema deve mostrar uma mensagem informando o usuário que ele deve escolher um objetivo para continuar(*MSG_01 - Vício não escolhido*);
6. O sistema retorna para o **FB_04;**

FE_02 - Obrigatoriedade de um nome para a tarefa

1. Este fluxo se inicia quando o usuário tenta prosseguir na criação de um meta com uma tarefa criada em branco;
2. O sistema deve mostrar uma mensagem informando o usuário acerca da meta em branco(*MSG_01 - Meta em branco*);
3. O sistema deve retornar para o **FB_04;**

FE_03 - Número digitado incorreto

1. Este fluxo se inicia quando o usuário coloca algum número negativonos seus objetivos ;
2. O sistema deve mostrar uma mensagem informando o usuário colocou algum informação incorreta (*MSG_02 - campos preenchidos incorretamente*);

2.6 Pós-condições

1. O sistema deve adicionar a meta a uma lista de metas criadas na tela principal do app.
2. O sistema computa os dados atuais de progresso do usuário na meta.

3 Caso de Uso

3.1 Identificação do caso de uso

UC_01 - Criar meta de financeira

3.2 Descrição do caso de uso

Este caso de uso está relacionado à criação de uma meta financeira. Portanto, por meio deste caso de uso, o usuário pode criar uma meta do tipo financeira. Uma meta financeira está relacionada às metas de economia de dinheiro, acumular um certo valor para comprar um produtor. Quaisquer metas que estejam relacionadas com a vida financeira das pessoas.

3.3 Atores envolvidos

1. Usuário

3.4 Pré-condições

1. O usuário deve estar com o aplicativo aberto, na tela principal.

3.5 Fluxos de Eventos

3.5.1 FB - Fluxo Básico

1. O usuário solicita a criação de uma meta;
2. O usuário informa qual o objetivo dele na meta(*FE_02 - Obrigatoriedade da escolha de um objetivo*);
3. O usuário informa qual a quantia atual já progredida;
4. O usuário informa qual a quantia estimada para a meta;(*FE_02 - Quantia estimada menor que quantia atual*)
5. O usuário informa as tarefas que ele deseja listar na sua meta;(*FE_03 - Obrigatoriedade de um nome para a tarefa*);
6. Usuário cria a meta;
7. O sistema adiciona a meta criada na lista de metas criadas da tela principal;
8. Caso de uso é encerrado;

3.5.2 Fluxos Alternativos

3.5.3 Fluxos de Exceção

FE_01 - Obrigatoriedade da escolha de um objetivo

1. Este fluxo se incia quando o usuário tenta prosseguir na criação da meta financeira sem setar o seu objetivo naquela meta;
2. O sistema deve mostrar uma mensagem informando o usuário que ele deve escolher um objetivo para continuar(*MSG_02 - Objetivo não escolhido*);
3. O sistema retorna para o **FB_02;**

FE_02 - Quantia estimada menor que quantia atual

1. Este fluxo se inicia quando o usuário tenta prosseguir na criação da meta financeira sem com um valor atual maior que o valor estimado;
2. O sistema deve mostrar uma mensagem informando o usuário que o valor atual é maior que o valor estimado para a meta(*MSG_02 - Valor atual maior que valor estimado*);
3. O sistema deve retornar para o **FB_03;**

FE_03 - Obrigatoriedade de um nome para a tarefa

1. Este fluxo se inicia quando o usuário tenta prosseguir na criação de um meta com uma tarefa criada em branco;
2. O sistema deve mostrar uma mensagem informando o usuário acerca da meta em branco(*MSG_01 - Meta em branco*);
3. O sistema deve retornar para o **FB_05;**

3.6 Pós-condições

1. O sistema deve adicionar a meta a uma lista de metas criadas na tela principal do app;
2. O sistema computa os dados atuais de progresso do usuário na meta;

4 Caso de Uso

4.1 Identificação do caso de uso

UC_02 - Criar meta de exercício

4.2 Descrição do caso de uso

Este caso de uso está relacionado a criação da meta de exercício , o usuário pode criar uma meta e tarefas para auxiliar em seu objetivo.Essa meta o usuário deve colocar os dias que ele deseja treinar e a hora do seu exercício

4.3 Atores envolvidos

2. usuário

4.4 Pré-condições

1 . usuário precisa estar na tela principal do aplicativo

4.5 Fluxos de Eventos

4.5.1 FB - Fluxo Básico

1. O usuário estar na tela principal
2. O usuário solicita a criação de meta;
3. O usuário clica na meta de exercício;
4. O usuário seleciona os dias da semana(*FE_01 - Obrigatoriedade da escolha dos dias da semana*);
5. O usuário seleciona os exercícios(*FE_02 - Obrigatoriedade da escolha dos exercícios*);
6. O usuário clica em adicionar tarefas(*FE_03 - Obrigatoriedade de um nome para a tarefa*);
7. O usuário cria a meta;
8. O sistema adiciona a meta criada na lista de metas criadas da tela principal;
9. Caso de meta encerrado;

4.5.2 Fluxos Alternativos

4.5.3 Fluxos de Exceção

FE_01 - Obrigatoriedade da escolha dos dias da semana

1. Este fluxo se inicia quando o usuário tenta prosseguir na criação da meta de exercício sem escolher os dias;
2. O sistema deve mostrar uma mensagem informando o usuário que ele deve escolher pelo menos 1 dia na semana (*MSG_01 - dia não selecionado*);
3. O sistema retorna para o **FB_04;**

FE_03 - Obrigatoriedade de um nome para a tarefa

1. Este fluxo se inicia quando o usuário tenta prosseguir na criação de um meta com uma tarefa criada em branco;
2. O sistema deve mostrar uma mensagem informando o usuário acerca da meta em branco(*MSG_02 - Meta em branco*);
3. O sistema deve retornar para o **FB_04;**

4.6 Pós-condições

3. O sistema deve adicionar a meta a uma lista de metas criadas na tela principal do app;
4. O sistema computa os dados atuais de progresso do usuário na meta;

5 Caso de Uso

5.1 Identificação do caso de uso

UC_02 - Filtrar meta;

5.2 Descrição do caso de uso

Este caso de uso está relacionado a filtragem de metas , o usuário terá como filtrar as metas desejadas para facilitar encontrá-las (meta de rotina de exercício, vício, dieta, financeira).

5.3 Atores envolvidos

3. Usuário

5.4 Pré-condições

1. O usuário precisa estar na tela de metas

5.5 Fluxos de Eventos

5.5.1 FB_01 - Fluxo Básico

1. O usuário solicita o filtro da lista de metas;
2. O usuário seleciona o tipo de meta que ele quer que seja filtrado;
3. O sistema filtra a lista de metas criadas pelo tipo que o usuário selecionou (*FA_01 - Filtro com lista de metas vazias*);
4. Caso de uso encerrado;

5.5.2 Fluxos Alternativos

5.5.3 Fluxos de Exceção

FE_01 - Filtro com lista vazia de metas

1. Este fluxo alternativo se inicia quando o usuário tenta filtrar uma lista de metas vazia;

2. O sistema deve mostrar uma mensagem para o usuário informando que nenhuma meta foi encontrada;
3. O sistema retorna para o **FB - passo 1**.

5.6 Pós-condições

1. O sistema mostra uma lista de metas filtrada pelo tipo que o usuário selecionou;

6 Caso de Uso

6.1 Identificação do caso de uso

UC_02 - Cancelar meta.

6.2 Descrição do caso de uso

Este caso de uso está relacionado ao cancelamento da meta criada pelo usuário.

6.3 Atores envolvidos

1. Usuário

6.4 Pré-condições

1. O usuário precisa ter criado uma meta.

6.5 Fluxos de Eventos

6.5.1 FB_01 - Fluxo Básico

1. O usuário seleciona a meta que deseja cancelar da lista de metas criadas da tela principal;
2. O usuário solicita o cancelamento da meta;
3. O sistema envia uma mensagem solicitando a confirmação do cancelamento da meta;
4. O usuário confirma o cancelamento(*FA_01 - Cancelar o cancelamento*);
5. O sistema exclui a meta da lista de metas criadas;
6. Caso de uso é encerrado;

6.5.2 Fluxos Alternativos

FA_01 - Cancelar o cancelamento

1. Este fluxo cancela o cancelamento da meta;
2. Caso de uso é encerrado;

6.5.3 Fluxos de Exceção

6.6 Pós-condições

1. A lista de metas criadas da tela principal é atualizada sem a meta que foi cancelada;

7 Caso de Uso

7.1 Identificação do caso de uso

UC_02 -Finalizar meta.

7.2 Descrição do caso de uso

Este caso de uso está relacionado finalização da meta que já foi concluída pelo usuário;

7.3 Atores envolvidos

2. Usuário

7.4 Pré-condições

1. O usuário precisa ter criado uma meta.
2. O usuário deve ter concluído;

7.5 Fluxos de Eventos

FB_01 - Fluxo Básico

1. O usuário seleciona a meta que deseja finalizar da lista de metas criadas da tela principal;
7. O usuário solicita a finalização da meta;
8. O sistema envia uma mensagem solicitando a confirmação da finalização da meta;
9. O usuário confirma a finalização(*FA_01 - Cancelar a finalização*);
10. O sistema seta a meta como concluída;
11. O sistema exclui a meta da lista de metas criada;
12. O sistema adiciona a meta finalizada na lista de metas concluídas da tela de metas concluídas;
13. Caso de uso é encerrado;

7.5.1 Fluxos Alternativos

FA_01 - Cancelar a finalização

3. Este fluxo se inicia quando o usuário cancela a finalização da meta;
4. Caso de uso é encerrado;

7.5.2 Fluxos de Exceção

7.6 Pós-condições

2. A lista de metas criadas da tela principal é atualizada sem a meta que foi finalizada pelo usuário;
3. A lista de metas concluídas da tela de metas concluídas é atualizada com a meta recém finalizada;

8 Caso de Uso

8.1 Identificação do caso de uso

UC_02 - Atualizar o valor atual da meta financeira;

8.2 Descrição do caso de uso

Este caso de uso está relacionado a atualização que o usuário faz do valor atual da meta financeira, que representa o valor que o usuário progrediu até o momento em seu objetivo. Isso envolve tanto decrementar, quanto incrementar o valor;

8.3 Atores envolvidos

1. Usuário

8.4 Pré-condições

1. O usuário precisa ter criado uma meta financeira.

8.5 Fluxos de Eventos

FB_01 - Fluxo Básico

1. O usuário seleciona a meta que deseja atualizar o valor atual;
2. O usuário informa o valor para atualizar a meta:

- Valor;

3. O usuário solicita o incremento do valor atual(*FA_01 - Usuário solicita o decremento do valor atual*);
4. O sistema incrementa o valor atual com o valor informado pelo usuário(FE_01 - Obrigatoriedade do valor);
5. Caso de uso é encerrado;

8.5.1 Fluxos Alternativos

FA_01 - Usuário solicita o decremento do valor atual

1. Este fluxo se inicia quando o usuário solicita o decremento do valor atual;
2. O sistema decrementa o valor atual com o valor informado pelo usuário(FE_02 - Obrigatoriedade do valor);
3. Caso de uso é encerrado;

8.5.2 Fluxos de Exceção

FE_02 - Obrigatoriedade do valor

1. Este fluxo se inicia quando o usuário solicita o incremento do valor atual da meta financeira sem informar nenhum valor;
2. O sistema mostra uma mensagem informando que o campo do valor está vazio(MSG_01 - Valor não informado);
3. sistema retorna para o **FB - passo 2**;

FE_01 - Obrigatoriedade do valor

1. Este fluxo se inicia quando o usuário solicita o decremento do valor atual da meta financeira sem informar nenhum valor;
2. O sistema mostra uma mensagem informando que o campo do valor está vazio(MSG_01 - Valor não informado);
3. sistema retorna para o **FB - passo 2**;

8.6 Pós-condições

1. O valor atual da meta é atualizado com o valor informado pelo usuário;

9 Caso de Uso

9.1 Identificação do caso de uso

UC_02 - Reiniciar contador de tempo da meta de vício

9.2 Descrição do caso de uso

Este caso de uso está relacionado a reinicialização do contador do tempo de vício. Quando o usuário descumpriu com o objetivo da meta, ele terá a opção de reiniciar o contador de tempo da meta de vício;

9.3 Atores envolvidos

1. Usuário

9.4 Pré-condições

1. O usuário precisa ter criado uma meta vício.

9.5 Fluxos de Eventos

FB_01 - Fluxo Básico

1. O usuário seleciona a meta que deseja reiniciar o contador;
2. O usuário solicita a reinicialização do contador de tempo da meta;
3. O sistema solicita uma confirmação;
4. O usuário confirma a reinicialização;
5. O sistema reinicializa o contador de tempo da meta;
6. Caso de uso é encerrado;

9.5.1 Fluxos Alternativos

FA_01 - Cancelamento de reinicialização

1. Esse fluxo se inicia quando o usuário cancela a confirmação da reinicialização do contador de tempo da meta;
2. Caso de uso é encerrado;

9.5.2 Fluxos de Exceção

FE_01 - Obrigatoriedade do valor

9.6 Pós-condições

1. O contador é reiniciado a partir de 0;

10 Caso de Uso

10.1 Identificação do caso de uso

UC_02 - Marcar tarefa como concluída

10.2 Descrição do caso de uso

Este caso de uso está relacionado a conclusão da tarefa realizada na meta.

Quando o usuário concluir alguma tarefa que ele adicionou à meta, ele terá a opção de marcar a checkbox da tarefa para saber que já a concluiu.

10.3 Atores envolvidos

1. Usuário

10.4 Pré-condições

1. O usuário precisa ter criado uma meta;
2. O usuário precisa ter adicionado uma tarefa a meta;

10.5 Fluxos de Eventos

FB_01 - Fluxo Básico

1. O usuário seleciona a meta que cuja tarefa foi concluída;
2. O usuário solicita que a tarefa seja marcada como concluída(FA_01 - Desmarcar tarefa);
3. O sistema seta a tarefa como concluída;
4. Caso de uso é encerrado;

10.5.1 Fluxos Alternativos

FA_01 - Desmarcar tarefa

1. Esse fluxo se inicia quando o usuário solicita que uma tarefa seja desmarcada;
2. O sistema desmarca a tarefa selecionada;
3. Caso de uso encerrado.

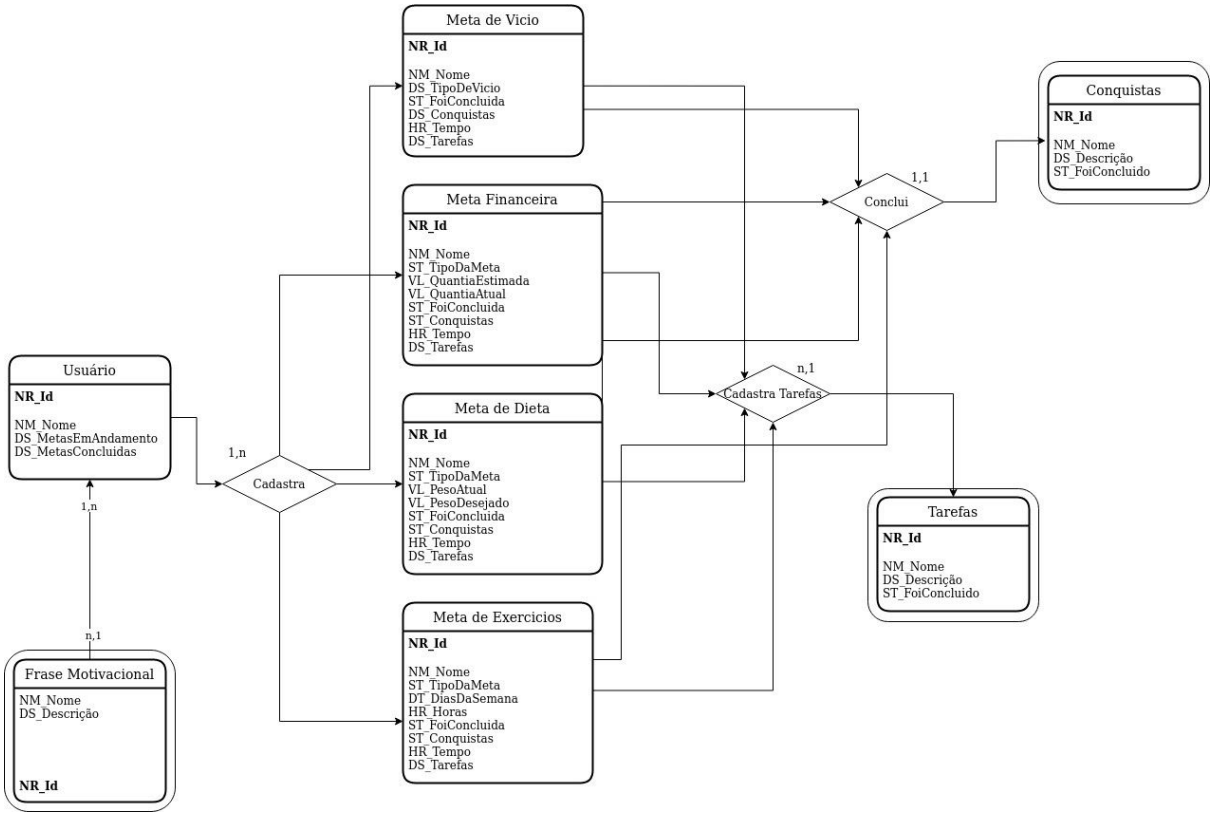
10.5.2 Fluxos de Exceção

10.6 Pós-condições

1. A meta deve estar marcada ou desmarcada.

4. Modelos Entidade Relacionamento (MER)

4.1 Modelo Lógico



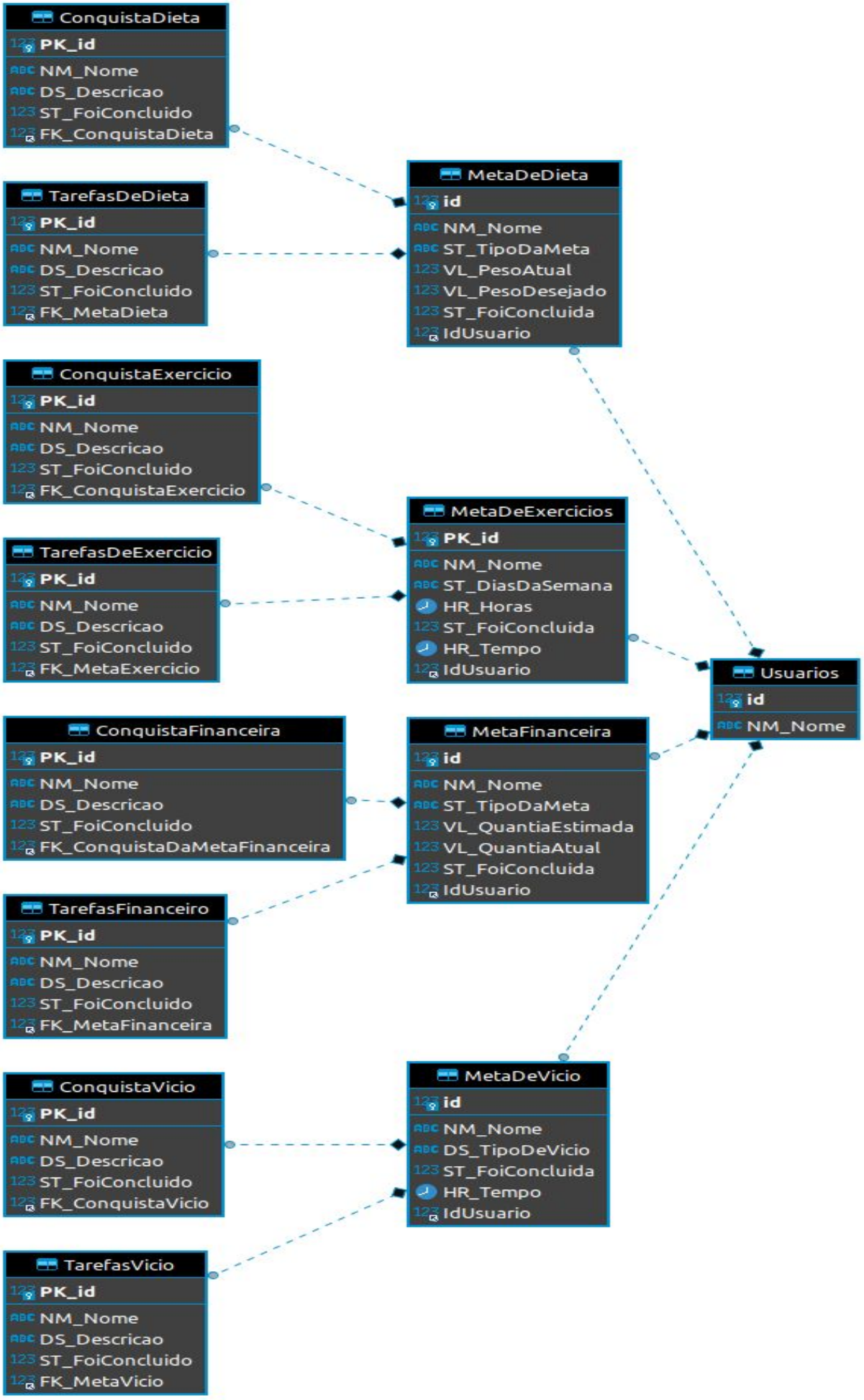
4.2 Modelo Físico

```
create database CanDoIt;
use CanDoIt;

CREATE TABLE Usuario(
NM_Nome varchar(20) NOT NULL,

id int primary key auto_increment not null,
```

```
NR_vicio int,
NR_dieta int,
NR_financeira int,
NR_exercicio int,

foreign key (NR_vicio) references MetaDeVicio(id),
foreign key (NR_dieta) references MetaFinanceira(id),
foreign key (NR_financeira) references MetaDeDieta(id),
foreign key (NR_exercicio) references
MetaDeExercicios(id)



);



CREATE TABLE MetaDeVicio(
NM_Nome varchar(20) NOT NULL,
DS_TipoDeVicio varchar(12),
ST_FoiConcluida boolean,
HR_Tempo datetime(3),



id int primary key auto_increment not null,
NR_Tarefa int,
NR_Conquista int,
foreign key (NR_Tarefa) references Tarefas(PK_id),
foreign key (NR_Conquista) references Conquistas(PK_id)
);

CREATE TABLE MetaFinanceira(
NM_Nome varchar(20) NOT NULL,
ST_TipoDaMeta char (10),
VL_QuantiaEstimada double,
```

```
VL_QuantiaAtual double,
ST_FoiConcluida boolean,
HR_Tempo datetime(3),


id int primary key auto_increment not null,
NR_Tarefa int,
NR_Conquista int,
foreign key (NR_Tarefa) references Tarefas(PK_id),
foreign key (NR_Conquista) references Conquistas(PK_id)
);

CREATE TABLE MetaDeDieta(
NM_Nome varchar(20) NOT NULL,
ST_TipoDaMeta char (10),
VL_PesoAtual double,
VL_PesoDesejado double,
ST_FoiConcluida boolean,
HR_Tempo datetime(3),

id int primary key auto_increment not null,
NR_Tarefa int,
NR_Conquista int,
foreign key (NR_Tarefa) references Tarefas(PK_id),
foreign key (NR_Conquista) references Conquistas(PK_id)
);

CREATE TABLE MetaDeExercicios(
NM_Nome varchar(20) NOT NULL,
ST_TipoDaMeta char (10),
ST_DiasDaSemana char(7),
HR_Horas time(2),
ST_FoiConcluida boolean,
```

```
HR_Tempo datetime(3),

id int primary key auto_increment not null,
NR_Tarefa int,
NR_Conquista int,
foreign key (NR_Tarefa) references Tarefas(PK_id),
foreign key (NR_Conquista) references Conquistas(PK_id)
);

CREATE TABLE Tarefas(
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,

PK_id int primary key auto_increment not null
);

CREATE TABLE Conquistas(
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,

PK_id int primary key auto_increment not null
);
```


ConquistaDieta
PK_id
NM_Nome
DS_Descricao
ST_FoiConcluido
FK_ConquistaDieta
TarefasDeDieta
PK_id
NM_Nome
DS_Descricao
ST_FoiConcluido
FK_MetaDieta
MetaDeDieta
id
NM_Nome
ST_TipoDaMeta
VL_PesoAtual
VL_PesoDesejado
ST_FoiConcluida
IdUsuario
ConquistaExercicio
PK_id
NM_Nome
DS_Descricao
ST_FoiConcluido
FK_ConquistaExercicio
TarefasDeExercicio
PK_id
NM_Nome
DS_Descricao
ST_FoiConcluido
FK_MetaExercicio
MetaDeExercicios
PK_id
NM_Nome
ST_DiasDaSemana
HR_Horas
ST_FoiConcluida
HR_Tempo
IdUsuario
Usuarios
id
NM_Nome
ConquistaFinanceira
PK_id
NM_Nome
DS_Descricao
ST_FoiConcluido
FK_ConquistaDaMetaFinanceira
MetaFinanceira
id
NM_Nome
ST_TipoDaMeta
VL_QuantiaEstimada
VL_QuantiaAtual
ST_FoiConcluida
IdUsuario
TarefasFinanceiro
PK_id
NM_Nome
DS_Descricao
ST_FoiConcluido
FK_MetaFinanceira
ConquistaVicio
PK_id
NM_Nome
DS_Descricao
ST_FoiConcluido
FK_ConquistaVicio
MetaDeVicio
id
NM_Nome
DS_TipoDeVicio
ST_FoiConcluida
HR_Tempo
IdUsuario
TarefasVicio
PK_id
NM_Nome
DS_Descricao
ST_FoiConcluido
FK_MetaVicio

3.1 Descrição Geral do Sistema Implementado

Utilizamos o padrão de nomenclatura dos atributos em cada uma das entidade, especificando qual o tipo dele, modelamos o banco de acordo a entidade usuário (principal entidade do sistema), para isso precisamos criar para uma tabela para cada um dos tipos de meta.

```
CREATE TABLE Usuario(
NM_Nome varchar(20) NOT NULL,

id int primary key auto_increment not null,
NR_vicio int,
NR_dieta int,
NR_financeira int,
NR_exercicio int,

foreign key (NR_vicio) references MetaDeVicio(id),
foreign key (NR_dieta) references MetaFinanceira(id),
foreign key (NR_financeira) references MetaDeDieta(id),
foreign key (NR_exercicio) references
MetaDeExercicios(id)



);

CREATE TABLE MetaDeVicio(
NM_Nome varchar(20) NOT NULL,
DS_TipoDeVicio varchar(12),
ST_FoiConcluida boolean,
HR_Tempo datetime(3),
```

```
id int primary key auto_increment not null,
NR_Tarefa int,
NR_Conquista int,
foreign key (NR_Tarefa) references Tarefas(PK_id),
foreign key (NR_Conquista) references Conquistas(PK_id)
);

CREATE TABLE MetaFinanceira(
NM_Nome varchar(20) NOT NULL,
ST_TipoDaMeta char (10),
VL_QuantiaEstimada double,
VL_QuantiaAtual double,
ST_FoiConcluida boolean,
HR_Tempo datetime(3),


id int primary key auto_increment not null,
NR_Tarefa int,
NR_Conquista int,
foreign key (NR_Tarefa) references Tarefas(PK_id),
foreign key (NR_Conquista) references Conquistas(PK_id)
);

CREATE TABLE MetaDeDieta(
NM_Nome varchar(20) NOT NULL,
ST_TipoDaMeta char (10),
VL_PesoAtual double,
VL_PesoDesejado double,
ST_FoiConcluida boolean,
HR_Tempo datetime(3),

id int primary key auto_increment not null,
NR_Tarefa int,
```

```
NR_Conquista int,
foreign key (NR_Tarefa) references Tarefas(PK_id),
foreign key (NR_Conquista) references Conquistas(PK_id)
);

CREATE TABLE MetaDeExercicios(
NM_Nome varchar(20) NOT NULL,
ST_TipoDaMeta char (10),
ST_DiasDaSemana char(7),
HR_Horas time(2),
ST_FoiConcluida boolean,
HR_Tempo datetime(3),

id int primary key auto_increment not null,
NR_Tarefa int,
NR_Conquista int,
foreign key (NR_Tarefa) references Tarefas(PK_id),
foreign key (NR_Conquista) references Conquistas(PK_id)
);
```

E cada meta terá, respectivamente, uma lista de tarefas que podem ser adicionadas. Colocamos um boolean para identificar se alguma já fora com concluida.

```
CREATE TABLE Tarefas(
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,

PK_id int primary key auto_increment not null
);
```

E também criamos uma tabela para armazenar todas as conquistas (que serão desbloqueadas caso o usuários atinja determinado objetivo , dado pela descrição do mesmo).

```
CREATE TABLE Conquistas(
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,

PK_id int primary key auto_increment not null
);
```

3.2 Problemas e soluções

Os problemas encontrados na parte da implementação estava relacionado em como iríamos organizar as relações entre as tabelas, como por exemplo, se cada lista de conquista(vício, financeira, exercício, dieta) iria ser uma entidade, então optamos por deixar cada tipo de dado em tabelas separadas assim ficando mais legível e melhor de aplicar . Na parte da modelagem encontramos dificuldades relação a quais tabelas iriamos criar e como seria uma melhor ligação entre todas as tabelas. Encontramos a solução através da análise de qual seria o melhor caminho, o mais rapido e economico para o sistema, assim tentando deixar o banco limpo e otimizado.

3.3 Funcionalidades implementadas

```
#selecionando todo de cada tabela
select * from Usuarios;
select * from MetaDeVicio;
select * from MetaDeDieta;
select * from MetaDeExercicios;
select * from MetaFinanceira;
select * from TarefasVicio;
select * from TarefasDeDieta;
select * from TarefasFinanceiro;
```

```
select * from TarefasDeExercicio;
select * from ConquistaVicio;
select * from ConquistaDieta;
select * from ConquistaFinanceira;
select * from ConquistaExercicio;
######


# criando um novo usuario
insert into Usuarios (NM_nome)
    values ("cleyson");

describe MetaFinanceira; # observar todas as colunas que
existem na tabela

# criando uma nova meta financeira
insert      into      MetaFinanceira      (NM_Nome      ,
ST_TipoDaMeta,VL_QuantiaEstimada,
VL_QuantiaAtual, ST_FoiConcluida , IdUsuario)
values ("ganhar money" , "acumular" , 100.00 , 1000.00
,false , 1 );



select * from MetaFinanceira; # verificando se foi
adicionado corretamente

describe TarefasFinanceiro; # observar todas as colunas
que existem na tabela

#criando uma tarefa para ajudar na meta financeira
insert into TarefasFinanceiro( NM_Nome , DS_Descricao,
ST_FoiConcluido,FK_MetaFinanceira )
```

```
values ("trabalhar mais", "trabaiar pq video games  tao caros " , true , 2 );

select * from TarefasFinanceiro; # observar se foi criado corretamente

# retorna o nome do usuario e nome , tipo e status das metas financeiras que ele criou
select Usuarios.NM_Nome , MetaFinanceira.NM_Nome , MetaFinanceira.ST_TipoDaMeta
,MetaFinanceira.ST_FoiConcluida
from Usuarios join MetaFinanceira
on Usuarios.id = MetaFinanceira.IdUsuario;


#retona todas as tarefas das metas financeiras ja criadas
select  *
from MetaFinanceira join TarefasFinanceiro
on MetaFinanceira.id = TarefasFinanceiro.FK_MetaFinanceira;


select * from TarefasFinanceiro
where TarefasFinanceiro.ST_FoiConcluido = false;

describe ConquistaFinanceira;

# inserindo uma conquista financeira
insert ConquistaFinanceira (NM_Nome , DS_descricao , ST_FoiConcluido , FK_ConquistaDaMetaFinanceira)
values ("50%", " voce atingiu 50% da sua meta" , false , 2);
```

```
select * from ConquistaFinanceira;

# retorna as conquistas cadastradas para cada meta
select ConquistaFinanceira.PK_id,
ConquistaFinanceira.NM_Nome , MetaFinanceira.id,
MetaFinanceira.NM_Nome ,ConquistaFinanceira.DS_Descricao
, ConquistaFinanceira.ST_FoiConcluido
from ConquistaFinanceira join MetaFinanceira
on ConquistaFinanceira.FK_ConquistaDaMetaFinanceira =
MetaFinanceira.id;


# altera o status da conquista deixando ela como
true(completa)
UPDATE ConquistaFinanceira SET ST_FoiConcluido='1' WHERE
PK_id='4';

# altera o status da tarefa deixando ela como
true(completa)
UPDATE TarefasFinanceiro SET ST_FoiConcluido='1' WHERE
PK_id='6';

# altera o status da meta financeira deixando ela como
true(completa)
UPDATE MetaFinanceira SET ST_FoiConcluida='1' WHERE
id='2';


# comandos para deletar uma meta que tenha conquista e
tarefas junto a ela , antes de excluir uma meta temos
#que excluir todas as ligaçoes que ela faz (chaves
estrangeiras)
```

```
delete from TarefasFinanceiro where FK_MetaFinanceira =
'2';
delete from ConquistaFinanceira where
FK_ConquistaDaMetaFinanceira = '2';
delete from MetaFinanceira where id = '2';


#
select * from MetaFinanceira;

#retorna todas as metas financeira em ordem crescente
select * from MetaFinanceira
order  by ST_FoiConcluida ;


#retorna todas as metas financeira em ordem decrescente
select * from MetaFinanceira
order  by ST_FoiConcluida desc ;



describe MetaDeVicio;
#criando meta de vicio
insert into MetaDeVicio (NM_Nome , DS_TipoDeVicio ,
ST_FoiConcluida , HR_Tempo, IdUsuario)
values ("parae de fumar" , "cigarro" , false , '00:00:00'
, 1)
;

select * from MetaDeVicio;


describe MetaDeDieta;
```

```
#criando meta de dieta
insert into MetaDeDieta (NM_Nome , ST_TipoDaMeta ,
VL_PesoAtual , VL_PesoDesejado, ST_FoiConcluida ,
IdUsuario)
values ("perde peso" , "emagrecer" , 110 ,90 , false ,1)
;
select * from MetaDeDieta;


describe MetaDeExercicios;

#criando meta de exercicio
insert into MetaDeExercicios (NM_Nome , ST_DiasDaSemana ,
HR_Horas , ST_FoiConcluida , IdUsuario)
values ("academia" , "2 4 6" , "20:00" , false ,1)
;
select * from MetaDeExercicios;


#retorna em uma linnha o nome de todas as metas
select Usuarios.id , MetaDeVicio.NM_Nome ,
MetaDeDieta.NM_Nome , MetaFinanceira.NM_Nome ,
MetaDeExercicios.NM_Nome
from Usuarios
 inner join MetaDeVicio on Usuarios.id =
MetaDeVicio.IdUsuario
 inner join MetaDeDieta on Usuarios.id =
MetaDeDieta.IdUsuario
 inner join MetaFinanceira on Usuarios.id =
MetaFinanceira.IdUsuario
 inner join MetaDeExercicios on Usuarios.id =
MetaDeExercicios.IdUsuario
```

;

4. Considerações finais

O projeto CanDoIt tem por objetivo auxiliar pessoas que sentem dificuldades em organizarem e seguirem motivadas em suas metas e objetivos. Em um mundo onde nosso dia a dia está cada vez mais atarefado, não encontramos tempo para pensarmos em nossas ambições e metas. O CanDoIt se propõe a auxiliá-las de forma simples e intuitiva, que essas pessoas mantenham-se firmes em suas metas, de forma que acompanhem de perto sua evolução, e sintam-se motivadas a ir mais longe. Ele também se propõe a auxiliar na quebra de problemas que as pessoas possam enfrentar para o cumprimento da meta, de forma que elas possam adicionar tarefas que as ajudem a concluir uma meta específica.

Em questão de resultados, conseguimos alcançar boa parte de nossas pretensões de desenvolvimento, pois é um sistema um tanto complexo, e nosso intuito é leva-lo para frente no futuro, refinando sua arquitetura e adicionando mais funcionalidades que possam auxiliar ainda mais as pessoas. Até o momento existem 4 possibilidades de metas: de vício, financeira, rotina de exercícios, e dieta para perda ou ganho de peso. Mais para frente buscaremos adicionar mais possibilidades para o usuário, bem como a possibilidade dele criar metas personalizadas a seu gosto.

Encontramos algumas dificuldades em relação a organização das relações das tabelas, e na quantidade de tabelas e quais tipos de tabelas o sistema necessitaria para comportar as entidades necessárias. A solução veio através de análise e discussões em grupo sobre qual seria o melhor caminho viável.

5. Bibliografia

HEUSER, Carlos. **Projeto de Banco de Dados**. 4. ed. [*S. l.*]: Editora Sagra Luzzatto, 1998. 192 p. v. 4.

APÊNDICE A - Código Fonte

```
create database CanDoIt; #criando o banco CanDoIt
use CanDoIt; #selecionando o banco




CREATE TABLE Usuarios ( #criaçao do entidade usuario
id smallint primary key auto_increment,
NM_Nome varchar(20) NOT NULL

);



CREATE TABLE MetaDeVicio( # criaçao da meta de vicio
NM_Nome varchar(20) NOT NULL,
DS_TipoDeVicio varchar(12),
ST_FoiConcluida boolean,
HR_Tempo time,
```

```
IdUsuario smallint,
id int primary key auto_increment,

FOREIGN KEY (IdUsuario) REFERENCES Usuarios(id)
);

CREATE TABLE MetaFinanceira(# criaçao da meta financeira
NM_Nome varchar(20) NOT NULL,
ST_TipoDaMeta char(10) not null,
VL_QuantiaEstimada decimal (8,2),
VL_QuantiaAtual decimal (8,2),
ST_FoiConcluida boolean,
IdUsuario smallint,

id int primary key auto_increment ,
FOREIGN KEY (IdUsuario) REFERENCES Usuarios(id)

);

CREATE TABLE MetaDeDieta( # criaçao da meta dieta
NM_Nome varchar(20) NOT NULL,
ST_TipoDaMeta char (10),
VL_PesoAtual double,
VL_PesoDesejado double,
ST_FoiConcluida boolean,

IdUsuario smallint,
id int primary key auto_increment,


FOREIGN KEY (IdUsuario) REFERENCES Usuarios(id)
);
```

```
drop table MetaDeExercicios;
drop table TarefasDeExercicio;
drop table ConquistaExercicio;

CREATE TABLE MetaDeExercicios( # criaçao da meta de exercicio
NM_Nome varchar(20) NOT NULL,
ST_DiasDaSemana char(7),
HR_Horas varchar(5),
ST_FoiConcluida boolean,

IdUsuario smallint,
PK_id int primary key auto_increment,



FOREIGN KEY (IdUsuario) REFERENCES Usuarios(id)
);

CREATE TABLE TarefasVicio( # tarefas de cada meta
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_MetaVicio int,

PK_id int primary key auto_increment ,
foreign key(FK_MetaVicio) references MetaDeVicio(id)

);

CREATE TABLE TarefasDeDieta( # tarefas de cada meta
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_MetaDieta int,
```

```
PK_id int primary key auto_increment ,
foreign key(FK_MetaDieta) references MetaDeDieta(id)

);

CREATE TABLE TarefasFinanceiro( # tarefas de cada meta
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_MetaFinanceira int,

PK_id int primary key auto_increment ,
foreign key(FK_MetaFinanceira) references MetaFinanceira(id)

);

CREATE TABLE TarefasDeExercicio( # tarefas de cada meta
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_MetaExercicio int,

PK_id int primary key auto_increment ,
foreign key(FK_MetaExercicio) references
MetaDeExercicios(PK_id)

);


CREATE TABLE ConquistaVicio( # lista de todas as conquistas do
app
NM_Nome varchar(20) NOT NULL,
```

```
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_ConquistaVicio int ,

PK_id int primary key auto_increment,
foreign key(FK_ConquistaVicio) references MetaDeVicio(id)

);

CREATE TABLE ConquistaDieta( # lista de todas as conquistas do
app
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_ConquistaDieta int ,

PK_id int primary key auto_increment,
foreign key(FK_ConquistaDieta) references MetaDeDieta(id)

);


CREATE TABLE ConquistaFinanceira( # lista de todas as
conquistas do app
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_ConquistaDaMetaFinanceira int ,

PK_id int primary key auto_increment,
foreign key(FK_ConquistaDaMetaFinanceira) references
MetaFinanceira(id)
```

```
);
CREATE TABLE ConquistaVicio( # lista de todas as conquistas do
app
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_ConquistaVicio int ,

PK_id int primary key auto_increment,
foreign key(FK_ConquistaVicio) references MetaDeVicio(id)

);
CREATE TABLE ConquistaExercicio( # lista de todas as
conquistas do app
NM_Nome varchar(20) NOT NULL,
DS_Descricao varchar(100),
ST_FoiConcluido boolean,
FK_ConquistaExercicio int ,

PK_id int primary key auto_increment,
foreign key(FK_ConquistaExercicio) references
MetaDeExercicios(PK_id)

);




###############


#selecionando todo de cada tabela
select * from Usuarios;
```

```
select * from MetaDeVicio;
select * from MetaDeDieta;
select * from MetaDeExercicios;
select * from MetaFinanceira;
select * from TarefasVicio;
select * from TarefasDeDieta;
select * from TarefasFinanceiro;
select * from TarefasDeExercicio;
select * from ConquistaVicio;
select * from ConquistaDieta;
select * from ConquistaFinanceira;
select * from ConquistaExercicio;
######


# criando um novo usuario
insert into Usuarios (NM_nome)
    values ("cleyson");

describe MetaFinanceira; # observar todas as colunas que
existem na tabela

# criando uma nova meta financeira
insert into MetaFinanceira (NM_Nome ,
ST_TipoDaMeta,VL_QuantiaEstimada,
VL_QuantiaAtual, ST_FoiConcluida , IdUsuario)
values ("ganhar money" , "acumular" , 100.00 , 1000.00 ,false
, 1 );



select * from MetaFinanceira; # verificando se foi adicionado
corretamente
```

```
describe TarefasFinanceiro; # observar todas as colunas que
existem na tabela

#criando uma tarefa para ajudar na meta financeira
insert into TarefasFinanceiro( NM_Nome , DS_Descricao,
ST_FoiConcluido,FK_MetaFinanceira )
values ("trabalhar mais", "trabaiar pq video games  tao caros
" , true , 2 );

select * from TarefasFinanceiro; # observar se foi criado
corretamente

# retorna o nome do usuario e nome , tipo e status das metas
financeiras que ele criou
select Usuarios.NM_Nome , MetaFinanceira.NM_Nome ,
MetaFinanceira.ST_TipoDaMeta ,MetaFinanceira.ST_FoiConcluida
from Usuarios join MetaFinanceira
on Usuarios.id = MetaFinanceira.IdUsuario;


#retona todas as tarefas das metas financeiras ja criadas
select  *
from MetaFinanceira join TarefasFinanceiro
on MetaFinanceira.id = TarefasFinanceiro.FK_MetaFinanceira;


select * from TarefasFinanceiro
where TarefasFinanceiro.ST_FoiConcluido = false;

describe ConquistaFinanceira;

# inserindo uma conquista financeira
```

```
insert ConquistaFinanceira (NM_Nome , DS_descricao ,
ST_FoiConcluido , FK_ConquistaDaMetaFinanceira)
values ("50%", " voce atingiu 50% da sua meta" , false , 2);

select * from ConquistaFinanceira;

# retorna as conquistas cadastradas para cada meta
select  ConquistaFinanceira.PK_id, ConquistaFinanceira.NM_Nome
, MetaFinanceira.id, MetaFinanceira.NM_Nome
,ConquistaFinanceira.DS_Descricao ,
ConquistaFinanceira.ST_FoiConcluido
from ConquistaFinanceira join MetaFinanceira
on ConquistaFinanceira.FK_ConquistaDaMetaFinanceira =
MetaFinanceira.id;


# altera o status da conquista deixando ela como
true(completa)
UPDATE ConquistaFinanceira SET ST_FoiConcluido='1' WHERE
PK_id='4';

#  altera o status da tarefa deixando ela como true(completa)
UPDATE TarefasFinanceiro SET ST_FoiConcluido='1' WHERE
PK_id='6';

#  altera o status da meta financeira deixando ela como
true(completa)
UPDATE MetaFinanceira SET ST_FoiConcluida='1' WHERE id='2';


# comandos para deletar uma meta que tenha conquista e tarefas
junto a ela , antes de excluir uma meta temos
```

```
#que excluir todas as ligaçoes que ela faz (chaves
estrangeiras)
delete from TarefasFinanceiro where FK_MetaFinanceira = '2';
delete from ConquistaFinanceira where
FK_ConquistaDaMetaFinanceira = '2';
delete from MetaFinanceira where id = '2';


#
select * from MetaFinanceira;

#retorna todas as metas financeira em ordem crescente
select * from MetaFinanceira
order  by ST_FoiConcluida ;


#retorna todas as metas financeira em ordem decrescente
select * from MetaFinanceira
order  by ST_FoiConcluida desc ;



describe MetaDeVicio;
#criando meta de vicio
insert into MetaDeVicio (NM_Nome , DS_TipoDeVicio ,
ST_FoiConcluida , HR_Tempo, IdUsuario)
values ("parae de fumar" , "cigarro" , false , '00:00:00' , 1)
;

select * from MetaDeVicio;


describe MetaDeDieta;
```

```
#criando meta de dieta
insert into MetaDeDieta (NM_Nome , ST_TipoDaMeta ,
VL_PesoAtual , VL_PesoDesejado, ST_FoiConcluida , IdUsuario)
values ("perde peso" , "emagrecer" , 110 ,90 , false ,1)
;
select * from MetaDeDieta;


describe MetaDeExercicios;

#criando meta de exercicio
insert into MetaDeExercicios (NM_Nome , ST_DiasDaSemana ,
HR_Horas , ST_FoiConcluida , IdUsuario)
values ("academia" , "2 4 6" , "20:00" , false ,1)
;
select * from MetaDeExercicios;


#retorna em uma linnha o nome de todas as metas
select  Usuarios.id , MetaDeVicio.NM_Nome ,
MetaDeDieta.NM_Nome , MetaFinanceira.NM_Nome ,
MetaDeExercicios.NM_Nome
from Usuarios
    inner join MetaDeVicio on Usuarios.id =
MetaDeVicio.IdUsuario
    inner join MetaDeDieta  on Usuarios.id =
MetaDeDieta.IdUsuario
    inner join MetaFinanceira on Usuarios.id =
MetaFinanceira.IdUsuario
    inner join MetaDeExercicios on Usuarios.id =
MetaDeExercicios.IdUsuario

;
```