



Universidade do Minho

Mestrado Integrado em Engenharia Informática

Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Letivo de 2020/2021

ReFood

Carlos Ferreira, Joel Martins, Manuel Moreira, Sara Dias

Dezembro, 2020

BD

Data de Receção	
Responsável	
Avaliação	
Observações	

ReFood

Carlos Ferreira, Joel Martins, Manuel Moreira, Sara Dias

Dezembro, 2020

Resumo

Este relatório foi desenvolvido no âmbito da disciplina de Bases de Dados para a associação ReFood, que foca a sua atividade na luta contra o desperdício de bens alimentares e contra a fome. Sendo uma organização sem fins lucrativos, confia o seu sucesso à disponibilidade e boa vontade de voluntários, que desempenham funções como a recolha de alimentos das fontes e a distribuição destes pelos beneficiários. A base de dados que iremos implementar tem como objetivo ajudar a gerir a informação referente a todo o processo que envolve a entrega de bens alimentares a pessoas mais carenciadas, desde a disposição de cada voluntário para executar determinado tipo de tarefa, às parcerias com os estabelecimentos que doam os alimentos, à gestão do tipo de comida que pode ser acumulada por data, passando pela gestão dos equipamentos necessários para que tudo seja exequível.

Ao longo deste relatório, são apresentadas na íntegra todas as etapas do desenvolvimento desta base de dados até à implementação física da mesma, pronta a ser utilizada.

Em resumo, numa primeira parte foi necessário estudar a situação que envolve o voluntariado, tal como a motivação, o contexto, as condições, e os objetivos, de modo a conseguirmos fazer uma análise sobre a utilidade e a viabilidade económica da implementação de uma base de dados neste contexto.

Numa fase posterior, através de metodologias estudadas na disciplina, foi necessário, recorrendo a vários métodos, levantar requisitos para a base de dados. De seguida, criamos um modelo concetual usando o brModelo e depois deduzimos o modelo lógico, respeitando as regras do mapeamento ER na ferramenta MySQL. Tudo isto foi realizado com a aprovação e validação por parte do cliente (gestores da ReFood), disposto sempre a mudanças. Resultando da acumulação de todas as etapas, foi finalmente praticável a implementação física da base de dados.

Concluindo, seguindo de forma geral as etapas descritas, foi possível a criação de uma base de dados útil e viável para o cliente, numa forma eficaz, segura e organizada.

Área de Aplicação: Desenho, Arquitetura, Desenvolvimento e Implementação de Sistemas de Bases de dados.

Palavras-Chave: Base de dados, Analise de requisitos, Entidades, Atributos, Relacionamento, Modelo conceptual, Modelo lógico, Normalização, Interrogações, *Triggers*, Índices, *Views*, *MySQL*, *MySQL WorkBench*, brModelo.

Índice

Resumo	i
Índice	ii
Índice de Figuras	iv
Índice de Tabelas	v
1 Definição do Sistema	1
1.1 Contexto de aplicação do sistema	1
1.2 Apresentação do Caso de Estudo	1
1.3 Fundamentação da implementação da base de dados	2
1.4 Análise da viabilidade do processo	2
1.5 Estrutura do Relatório	3
2 Levantamento e Análise de Requisitos	3
2.1 Método de levantamento e de análise de requisitos adotado	3
2.2 Requisitos levantados	4
2.2.1 Requisitos de descrição	4
2.2.2 Requisitos de exploração	4
2.2.3 Requisitos de controlo	4
2.3 Análise e validação geral dos requisitos	5
3 Modelação conceptual	5
3.1 Apresentação da abordagem de modelação realizada	5
3.2 Identificação e caracterização das entidades	5
3.3 Identificação e caracterização dos relacionamentos	7
3.4 Identificação e caracterização da associação dos atributos com as entidades e relacionamentos	12
3.5 Apresentação e explicação do diagrama ER	14
3.6 Validação do modelo de dados produzido	14
4 Modelação Lógica	15
4.1 Construção e validação do modelo de dados lógico	15
4.1.1 Relacionamentos de 1 para N	15
4.1.2 Relacionamentos de N para N	15
4.2 Desenho do modelo lógico	17
4.3 Validação do modelo através da normalização	17
4.4 Validação do modelo com interrogações do utilizador	18
4.5 Revisão do modelo lógico produzido	19
5 Implementação Física	20
5.1 Seleção do sistema de gestão de base de dados	20
5.2 Tradução do sistema lógico para sistema de gestão de bases de dados escolhido em SQL	20
5.3 Tradução das interrogações do utilizador para SQL	20

5.3.1	Alguns Exemplos	20
5.4	Escolha, definição e caracterização de índices em SQL	22
5.5	<i>Triggers</i> MySQL	22
5.6	Estimativa do espaço em disco de base de dados e taxa de crescimento anual	23
5.7	Definição e caracterização das vistas de utilização em SQL	24
5.8	Revisão Do Sistema Implementado	25
6	Conclusões	25
	Anexos	27
I.	Anexo 1 – Script de Inicialização da Base de Dados	27
II.	Anexo 2 – Script de População Inicial	31
III.	Anexo 3 – Scripts das queries	36
IV.	Anexo 4 – Scripts das Views, Triggers e Indexes	38

Índice de Figuras

Figura 1 - Relacionamento Centro de Operações - Voluntário	7
Figura 2 - Relacionamento Centro de Operações – Beneficiário	8
Figura 3 - Relacionamento Centro de Operações – Alimento	8
Figura 4 - Relacionamento Centro de Operações – Fonte	9
Figura 5 - Relacionamento Fonte - Recolha	10
Figura 6 - Relacionamento Voluntário – Recolha	10
Figura 7 - Relacionamento Alimento – Recolha	11
Figura 8 - Relacionamento Beneficiário – Alimento	11
Figura 9 - Modelo conceptual desenvolvido	14
Figura 10 - Modelo lógico desenvolvido	17
Figura 11 - Código obtido na Query 1	20
Figura 12 - Código obtido na Query 2	21
Figura 13 - Código obtido na Query 3	21
Figura 14 - Código obtido na Query 4	21
Figura 15 - Código obtido na Query 5	22
Figura 16 - Índices adicionados	22
Figura 17 - <i>Triggers</i> criados	23
Figura 18 - Vistas adicionadas	24

Índice de Tabelas

Tabela 1 - Espaço em disco esperado por entrada

23

1 Definição do Sistema

1.1 Contexto de aplicação do sistema

Com o decorrer do tempo, os Sistemas de Gestão de Base de Dados (SGDB) tornaram-se o desenvolvimento mais importante da área da engenharia de Software, sendo totalmente fundamentais no decorrer de diversas atividades, atualmente.

A área da solidariedade tem tido cada vez mais um impacto importante na nossa sociedade. Na atualidade, surgem estudos feitos pelas Nações Unidas que relatam que, em todo o mundo, cerca de um terço dos alimentos produzidos são desperdiçados, o que se traduz no facto de serem perdidos no percurso entre o produtor e o consumidor, adicionado ainda ao desperdício pelo qual cada consumidor é responsável. Só em Portugal, cerca de um milhão de toneladas de comida em bom estado acaba, todos os anos, no lixo. Estes estudos demonstram uma situação muito grave, que indica que, atualmente, algo jamais visto ao longo da história, os humanos produzem mais do que o que consomem, sendo estas perdas significativas para o meio ambiente, onde existe desperdício de recursos como a água, energia e mão de obra.

Tal pode acontecer por vários fatores, entre eles problemas de requisitos do mercado convencional, como alimentos rejeitados pelos produtores por não possuírem a forma ou cor desejada, ou problemas na definição da data e validade. No entanto, os maiores responsáveis são os inúmeros consumidores que desperdiçam toneladas de comida porque compram mais do que o que conseguem consumir.

Reconhecendo a gravidade do problema, inúmeras soluções são impostas para tentar atenuar de alguma forma a situação, conjugando com o facto de que tantas pessoas não têm possibilidades financeiras de se alimentar convenientemente. Com base nesta filosofia, originou-se o movimento ReFood, que combate o desperdício alimentar nos restaurantes, pastelarias, padarias, cafés e supermercados, enquanto ajuda quem mais necessita.

1.2 Apresentação do Caso de Estudo

Uma forma muito fácil de conseguir uma visão geral do movimento ReFood e o seu objetivo, é visitar o website <http://www.re-food.org/pt>.

Em resumo, o projeto ReFood é um movimento de voluntariado criado em Portugal com o lema de aproveitar para alimentar. Convida todo o tipo de membros da sociedade desde voluntários a instituições e empresas, a participar no impacto social positivo e combater um dos maiores problemas que a sociedade enfrenta no momento – o desperdício de recursos. A ReFood atualmente recruta, organiza e mantém centenas de voluntários todos os dias para resgatar o excesso de comida de diferentes estabelecimentos e oferecer a quem realmente necessita, estritamente de boa vontade.

Consegue oferecer diariamente a muitas pessoas grandes quantidades de alimentos nutritivos em bom estado e sem custos significativos, isto é, quase a custo zero.

Apresenta bases do norte ao sul de Portugal tendo várias comunidades no Algarve, Beira Interior, Beira Litoral, Lisboa, Minho, Porto, Ribatejo, Sul Do Tejo e Zona Oeste – a lista completa está indicada no seu [website](#).

Espera-se que, no futuro, o movimento venha a aumentar ainda mais, espalhando a positividade em cada comunidade e agrupando cada vez mais pessoas.

1.3 Fundamentação da implementação da base de dados

A razão pela qual o projeto ReFood necessita de uma base de dados é, principalmente, devido à grande quantidade de alimentos que é distribuída todos os dias, o aumento significativo de voluntários e de fontes de alimentos. Estes aumentos inesperados originam desafios na gestão dos horários, turnos e funções de cada voluntário, e mostra-se necessário organizar os dados de forma mais competente, e como é bem sabido, a melhor alternativa é recorrer à tecnologia, aproveitando a sua capacidade de guardar, analisar e gerir a informação, rápida e eficazmente.

Para além disso, foi pedido que se desenvolvessem outras capacidades, como por exemplo o registo da quantidade de famílias e beneficiários ajudados, e a quantidade de alimentos reaproveitados, o que ajudaria na divulgação da associação e possível crescimento desta, de forma a colmatar mais desperdícios alimentares e a auxiliar mais famílias necessitadas.

1.4 Análise da viabilidade do processo

Sendo que o movimento ReFood não requer a adaptação de uma base de dados ou software pré-existente, a construção da respetiva base de dados será um pouco trabalhosa no processo de análise de requisitos e estudo da situação. Em contra partida, não sofrerá de problemas de compatibilidade, sendo permitido desde já criar algo robusto e durável, usando métodos com os quais estamos familiarizados.

Tem-se então como objetivo a implementação de uma base de dados relacional robusta e durável, propensa a futuras alterações, e que possibilita, tal como todas as bases de dados do género, o armazenamento metodológico de informação relacional, que permita ao cliente, de forma fácil e rápida, manusear essa informação para corresponder às suas necessidades, não afetando a integridade dos dados.

A criação da base de dados dará ao cliente a possibilidade de gerir a circulação de grandes quantidades de alimentos, gerir horários e equipamento, assim como guardar, para fins estatísticos, grandes quantidades de dados referentes à gestão da organização, que poderão ser usados para ajustar as metodologias.

Concluindo, a solução que pretendemos implementar irá permitir ao cliente uma melhor gestão do Centro de Operações e uma melhoria mais metódica dos serviços prestados em benefício da sociedade.

1.5 Estrutura do Relatório

Ao longo deste relatório são descritas todas as fases de desenvolvimento do nosso projeto.

Inicialmente começamos por definir e contextualizar o nosso problema, analisando as vantagens trazidas com a implementação de um SBD relacional. De seguida também definimos uma série de requisitos que concluímos serem importantes de serem implementados na BD de modo a que esta seja capaz de responder. Por último, partimos o desenvolvimento em diferentes fases: conceptual, lógica e física, validando e justificando cada uma delas.

Finalmente, analisamos o projeto de um modo global e realçamos algumas possíveis melhorias que podiam ter sido feitas.

2 Levantamento e Análise de Requisitos

2.1 Método de levantamento e de análise de requisitos adotado

Em geral, para a realização do levantamento de requisitos, foi necessária a recolha de bastante informação de origem em interações ativas com o movimento de voluntariado. Para tal, foram usadas algumas técnicas fact-finding para tentar perceber o funcionamento do sistema, mais propriamente a organização e descrição de entidades e as respetivas relações.

Algumas das técnicas utilizadas foram:

Entrevistas: foram feitas entrevistas a colaboradores voluntários, aos gestores de turno e da organização, e aos responsáveis pelos processos de recolha e distribuição de alimentos.

Questionário: foram feitas perguntas aos diferentes colaboradores, como por exemplo a descrição detalhada da sua função, dificuldades com que se deparam, e sugestão de melhorias.

Observação/interação: um dos membros deste grupo de trabalho participa ativamente na associação como voluntário, tendo recolhido do ponto de vista de engenheiro informático várias informações necessárias à criação da base de dados.

2.2 Requisitos levantados

2.2.1 Requisitos de descrição

1. O Centro de Operações é definido pelo seu código e endereço.
2. Quando um Voluntário se regista, terá de indicar o seu nome, contacto e horários em que pretende voluntariar-se.
3. Uma Fonte de alimentos terá de ter um nome, contacto, endereço e o horário em que está disponível para recolhas.
4. Um Alimento tem um nome, tipo e a sua quantidade.
5. Uma Recolha é feita por um Voluntário a uma Fonte e está associada a uma quantidade diferente de diferentes tipos de alimentos.
6. Uma Recolha precisa de uma data.
7. Cada Voluntário está associado a um Centro de Operações.
8. Um Centro de Operações tem armazenado uma certa quantidade de um tipo de Alimento.
9. Um Beneficiário precisa de um nome e contacto.
10. Um Beneficiário pode ter certas restrições alimentares.
11. Um Beneficiário registado no sistema pode, numa certa data, receber vários alimentos em diferentes quantidades, desde que estes estejam armazenados no Centro de Operações.
12. No registo de recolhas e entregas, a quantidade de Alimento presente no armazém deve ser atualizada.

2.2.2 Requisitos de exploração

1. O Voluntário pode verificar a lista de todos os alimentos armazenados.
2. O Beneficiário pode verificar a lista de entregas que recebeu, por data.
3. O Voluntário pode verificar a lista de recolhas que realizou, por data.
4. O Voluntário pode ver o horário de uma fonte.
5. O Voluntário pode ver todos os alimentos e quantidade recolhidos numa data.

2.2.3 Requisitos de controlo

1. A Fonte de alimentos pode alterar o seu contacto, endereço e horário.
2. O Voluntário pode alterar o seu contacto e horário.
3. O Beneficiário pode alterar o seu contacto e as suas restrições alimentares.
4. O Voluntário pode adicionar alimentos ao Centro de Operações e registar entregas e recolhas.
5. No registo de recolhas e entregas, a quantidade de Alimento presente no armazém do Centro de Operações deve ser atualizada

2.3 Análise e validação geral dos requisitos

Após o levantamento de requisitos, foi necessária uma reflexão sobre eles, o que levou a muitas discussões entre os elementos do grupo e os participantes na associação para aprovar os mesmos, tendo sido necessário dar grande importância ao nível de detalhe no qual se registava o fluxo das quantidades de um certo alimento.

Os dados referentes a cada entidade não demonstraram muita importância, mas as relações entre elas foram bastante discutidas, principalmente quando delas surgiu a importância de indicar quantidades ou datas.

Esta fase teve bastante relevância, servindo para chegar a uma conclusão sobre o objetivo da base de dados, e ao mesmo tempo com o bônus de facilitar imenso as próximas fases do desenvolvimento da base de dados, não sendo necessário rever ou modificar parâmetros, já que ficaram todos muito bem definidos.

3 Modelação conceptual

3.1 Apresentação da abordagem de modelação realizada

Inicialmente começamos por construir um modelo conhecido como Diagrama ER. Uma vez que este não é muito técnico, permite que a estrutura base da nossa base de dados possa ser facilmente compreendida.

Para construir o Diagrama ER, começamos por identificar as entidades, atributos e relacionamentos mais relevantes da análise dos requisitos e de seguida adicionamos estes ao diagrama. Por último, adicionamos também as cardinalidades de cada um dos relacionamentos.

Por fim, validamos se todos os requisitos do cliente estavam de alguma forma representados no modelo, de modo a assegurar que a futura base de dados seja coerente e satisfaça o que lhe é pedido.

3.2 Identificação e caracterização das entidades

Atendendo aos requisitos levantados, identificamos as seguintes entidades:

- **Centro de Operações**

O Centro de Operações é a base onde quase todo o processo se desenrola. É representado por um **id** e pelo **endereço**, constituído pelo **código postal**, a **rua**, a **localidade** e **id**.

- **Fonte**

A Fonte de alimentos é um estabelecimento de restauração que se disponibilizou para acabar com o desperdício alimentar, e é, portanto, o local onde cada voluntário vai fazer a recolha e reaproveitamento de algum tipo de alimento. É identificada por um **idFonte**, **nome** do estabelecimento, **contacto** (constituído por **e-mail**, número de **telefone** e **idContacto**), **endereço** (constituído pelo **código postal**, a **rua**, a **localidade** e **id**) e pelo **horário** (constituído pelo **início**, **fim**, **dia da semana** e **idHorario**) em que está disponível para serem efetuadas as recolhas dos excedentes alimentares.

- **Voluntário**

O Voluntário é a pessoa individual que se voluntaria para a causa do desperdício alimentar na ReFood. É identificado por um **idVoluntário**, **nome**, **contacto** (constituído por **e-mail** e número de **telemóvel** e **idContacto**) e horário (constituído pelo **início**, **fim**, **dia da semana** e **idHorario**) em que está disponível para realizar as funções necessárias no Centro de Operações.

- **Beneficiário**

O Beneficiário é o agregado familiar que recebe apoio da ReFood. É identificado por um **idBeneficiario**, **nome** (do representante do agregado), **contacto** (constituído por **e-mail** e número de **telemóvel** e **idContacto**), **número de elementos** que compõe o agregado familiar e eventuais **restrições alimentares** que algum constituinte da família possa apresentar.

- **Recolha**

A Recolha é o processo de levantar da Fonte os Alimentos disponibilizados por esta. É caracterizada por um **idRecolha** e pela **data** em que foi realizada por algum Voluntário.

- **Alimento**

O Alimento é a comida disponibilizada por alguma Fonte e que vai ser aproveitada para alimentar algum beneficiário. É caracterizado pelo seu **nome**, **tipo** (pão, bolos, fruta, alimentos confeccionados ou alimentos de reserva) e **quantidade**.

3.3 Identificação e caracterização dos relacionamentos

- **Relacionamento Centro de Operações – Voluntário**

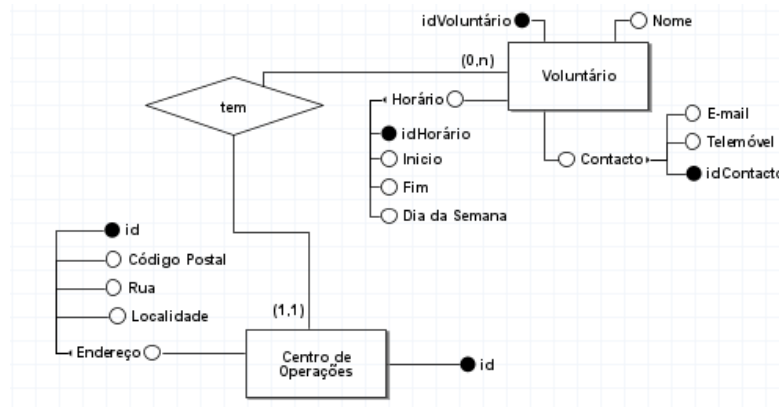


Figura 1 - Relacionamento Centro de Operações - Voluntário

Relacionamento: Centro de Operações tem Voluntário.

Descrição: Para conseguir funcionar, a associação ReFood conta com vários voluntários que realizem as tarefas necessárias. Desta forma, o Centro de Operações tem diversos Voluntários a trabalhar. É importante que a informação dos voluntários que constituem o CO seja armazenada, para se ter acesso aos registos das tarefas que cada um realiza e onde opera.

Cardinalidade: Centro de Operações (1,1) – Voluntário (0,n). Um determinado voluntário opera num e num só CO. No entanto, um CO pode possuir bastantes voluntários.

Atributos: Este relacionamento não tem atributos.

- **Relacionamento Centro de Operações – Beneficiário**

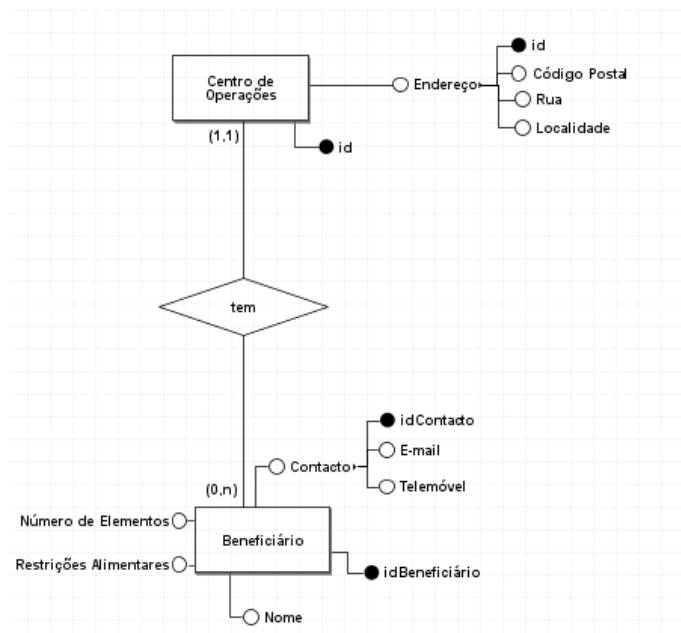


Figura 2 - Relacionamento Centro de Operações – Beneficiário

Relacionamento: Centro de Operações tem Beneficiário.

Descrição: O objetivo do projeto ReFood é, para além de evitar o desperdício de recursos alimentares, ajudar as pessoas mais necessitadas, dando-lhes refeições que se mostram, por vezes, os únicos alimentos que consomem ao longo do dia. Assim, sendo o Beneficiário das entidades principais do sistema, torna-se importante manter as informações referentes a cada agregado familiar, de forma a ser possível o controlo e acompanhamento dos alimentos entregues a cada um.

Cardinalidade: Centro de Operações (1,1) – Beneficiário (0,n). Um Beneficiário pode receber apoio apenas de um CO. No entanto, um CO funciona ajudando vários beneficiários.

Atributos: Este relacionamento não tem atributos.

- **Relacionamento Centro de Operações – Alimento**

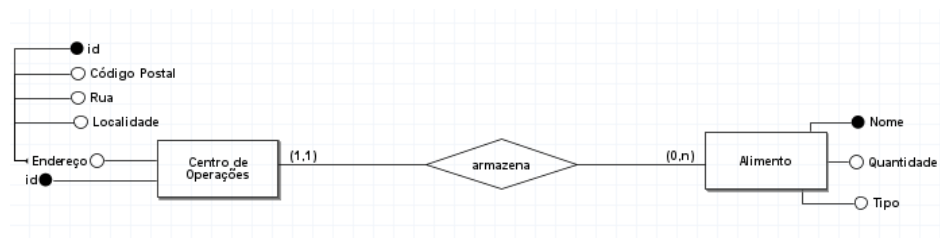


Figura 3 - Relacionamento Centro de Operações – Alimento

Relacionamento: Centro de Operações armazena Alimento.

Descrição: Para ser concretizado o objetivo de auxiliar famílias necessitadas, é necessário guardar a informação de cada alimento recolhido e armazenado no CO, de forma a ser possível o acompanhamento das recolhas e entregas de comida feitas, e gerir da melhor forma a quantidade de alimentos disponibilizados para cada agregado familiar.

Cardinalidade: Centro de Operações (1,1) – Alimento (0,n). Cada alimento está armazenado apenas num CO. No entanto, o CO pode ter vários alimentos armazenados, ou, em dias mais críticos onde não tenha havido excedentes, pode não ter nenhum alimento disponível para entregar às famílias.

Atributos: Este relacionamento não tem atributos.

- **Relacionamento Centro de Operações – Fonte**

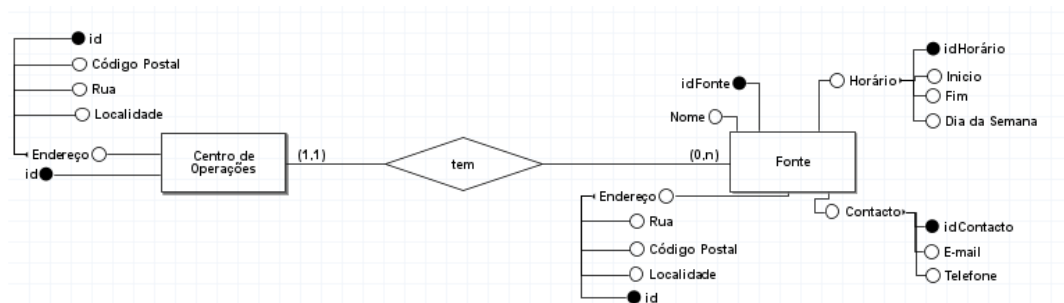


Figura 4 - Relacionamento Centro de Operações – Fonte

Relacionamento: Centro de Operações tem Fonte.

Descrição: As Fontes de alimentos são uma das entidades principais do sistema, pois são estas que tornam possível que haja alimentos e refeições preparadas para entregar a quem mais necessita. Deste modo, para que se mantenha uma boa gestão dos estabelecimentos de restauração que apoiam o projeto, é importante armazenar a informação de cada Fonte.

Cardinalidade: Centro de Operações (1,1) – Fonte (0,n). Cada fonte de alimentos disponibiliza os seus excedentes apenas a um CO. No entanto, para que o funcionamento seja maximizado, o CO necessita de ter várias fontes dispostas a ajudar.

Atributos: Este relacionamento não tem atributos.

- **Relacionamento Fonte – Recolha**

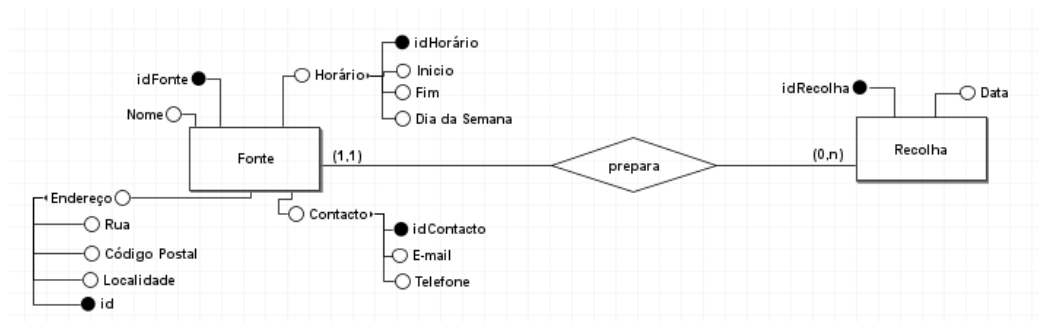


Figura 5 - Relacionamento Fonte - Recolha

Relacionamento: Fonte prepara Recolha.

Descrição: Para que seja possível a entrega dos alimentos aos beneficiários, é necessário que primeiro haja uma recolha de determinada Fonte. Da mesma forma, é importante armazenar a informação relativa a cada Recolha efetuada, para se manter o registo dos alimentos que entram no CO.

Cardinalidade: Fonte (1,1) – Recolha (0,n). Uma certa recolha que inclui alimentos é preparada apenas pela fonte que disponibilizada esses mesmos alimentos. No entanto, uma fonte pode preparar várias recolhas em diferentes dias ou fases do dia, e pode até não ter excedentes suficientes para preparar alguma.

Atributos: Este relacionamento não tem atributos.

- **Relacionamento Voluntário – Recolha**

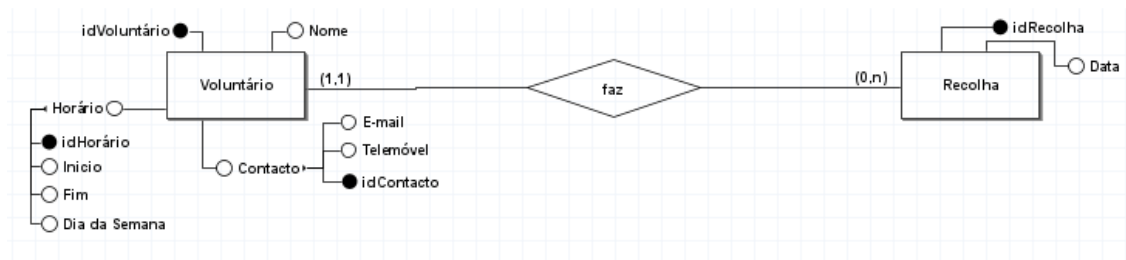


Figura 6 - Relacionamento Voluntário – Recolha

Relacionamento: Voluntário faz Recolha.

Descrição: Para que os alimentos sejam devidamente entregues aos beneficiários, é necessário que pelo menos um voluntário faça recolha de alimentos. É importante armazenar a informação de quem faz a recolha.

Cardinalidade: Voluntário (1,1) – Recolha (0,n). Uma recolha é feita por um e um só voluntário. No entanto, um determinado voluntário pode fazer várias ou nenhuma recolha, caso esteja encarregue de outras funções.

Atributos: Este relacionamento não tem atributos.

- **Relacionamento Alimento – Recolha**

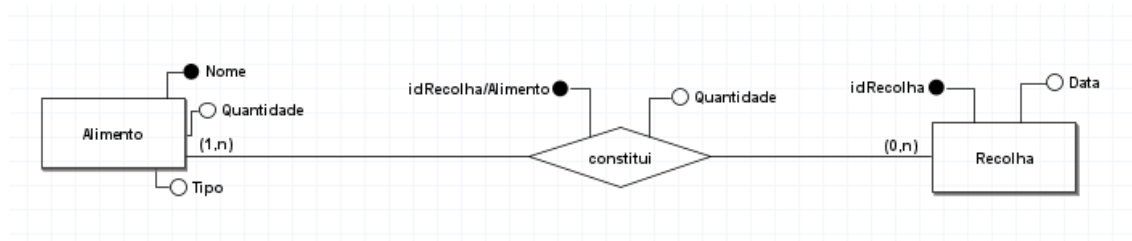


Figura 7 - Relacionamento Alimento – Recolha

Relacionamento: Alimento constitui Recolha.

Descrição: Intuitivamente, para que seja necessário fazer uma Recolha, é necessário que nela existam Alimentos. A informação sobre que alimentos constituem uma recolha é relevante, para se manter o registo da comida que entra no CO e é depois distribuída pelos beneficiários.

Cardinalidade: Alimento (1,n) – Recolha (0,n). Um determinado alimento pode não constar em nenhuma recolha, ou pode constar em várias, caso as fontes disponibilizem o mesmo tipo de comida. Já a recolha terá de ter obrigatoriamente pelo menos um alimento, podendo chegar a ter vários diferentes.

Atributos: Este relacionamento tem dois atributos: **quantidade** e **idRecolha/Alimento**. O primeiro representa a quantidade de alimentos diferentes que uma determinada recolha tem, e o segundo é o identificador da recolha que teve certo alimento.

- **Relacionamento Beneficiário – Alimento**

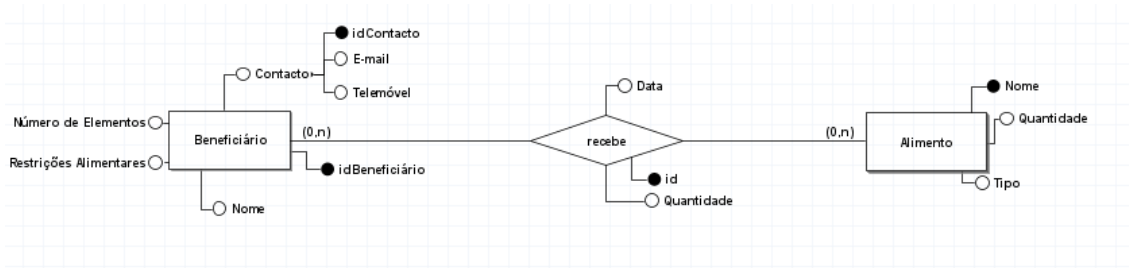


Figura 8 - Relacionamento Beneficiário – Alimento

Relacionamento: Beneficiário recebe Alimento.

Descrição: Naturalmente, para cumprir o propósito do projeto ReFood, os Beneficiários têm de receber Alimentos. Para que seja feita uma boa coordenação e gestão das refeições

disponibilizadas, é importante guardar a informação sobre o que cada agregado familiar recebe, principalmente atendendo ao número de elementos da família e às possíveis restrições alimentares que tenham. Esses dados são guardados atendendo à data da entrega e à quantidade entregue, informações importantes de registar para se acompanhar os processos, caso haja algum problema.

Cardinalidade: Beneficiário (0,n) – Alimento (0,n). Um beneficiário pode receber vários alimentos de uma vez, ou, quando não houve recolhas por falta de excedentes alimentares, pode não conseguir receber qualquer refeição nesse dia. Um determinado alimento pode ser entregue a nenhuma ou a várias famílias, dependendo da quantidade existente no CO desse mesmo alimento, e das necessidades de cada família, atendendo ao número de elementos do agregado.

Atributos: Este relacionamento possui dois atributos: **data**, **quantidade** e **id**. A data representa a altura em que foi feita a entrega de determinados alimentos a um determinado beneficiário, e a quantidade representa a quantidade de alimentos diferentes doados. O identificador representa a entrega que foi constituída por certo alimento.

3.4 Identificação e caracterização da associação dos atributos com as entidades e relacionamentos

- **Alimento**

Todos os atributos desta entidade (nome, quantidade, tipo) são simples e possuem um único valor, sendo o nome a chave primária, pois é o atributo que representa mais única, eficiente e eficazmente o Alimento. Em relação aos tipos de dados, o nome é um VARCHAR(45) não nulo, a quantidade é um INT não nulo e o tipo é um VARCHAR(45).

- **Beneficiário**

A entidade é constituída pelos atributos idBeneficiario (INT) não nulo, nome (VARCHAR(45)), número de elementos (INT) não nulo e restrições alimentares (VARCHAR(100)), que são simples e constituídos por um único valor. O contacto (INT) é um atributo composto constituído pela sua chave primária idContacto (INT) não nula, e-mail (VARCHAR(45)) e pelo telemóvel (VARCHAR(45)). A chave primária é o código que representa o beneficiário.

- **Voluntário**

O voluntário é constituído pela chave primária idVoluntário (INT) não nula e pelo atributo nome (VARCHAR(45)), que são ambos simples e possuem só um valor. O atributo horário (INT) não nulo é composto por início (INT) não nulo, fim (INT) não nulo, dia da semana (INT) não nulo e a chave primária idHorario (INT) não nula. Tal como no

beneficiário, é constituído pelo atributo contacto (INT), que é composto e que contém o e-mail (VARCHAR(45)), o telemóvel (VARCHAR(45)) e a chave primária idContacto (INT) não nula.

- **Recolha**

A recolha é constituída pela chave principal idRecolha (INT) não nula e ainda o atributo data (DATE). Ambos são simples e com apenas um valor.

- **Fonte**

A chave primária da fonte é o atributo idFonte (INT) não nulo. Este juntamente com o nome (VARCHAR(45)) são os dois atributos simples e que possuem só um valor. Para além disso, o atributo horário (INT) não nulo é composto pelo início (INT) não nulo, fim (INT) não nulo, dia da semana (INT) não nulo e a chave primária idHorario (INT) não nula. A fonte é constituída ainda pelo atributo composto contacto (INT), que contém o e-mail (VARCHAR(45)), o telefone (VARCHAR(45)) e a chave primária idContacto (INT) não nula; e pelo atributo composto endereço (INT) não nulo, que é constituído pela sua chave primária id (INT) não nula, rua (VARCHAR(45)), localidade (VARCHAR(45)) e código postal (VARCHAR(45)), todos simples e só com um valor.

- **Centro de operações**

O centro de operações é constituído pela chave primária id (INT) não nula, um atributo simples e só com um valor; e pelo atributo composto endereço (INT) não nulo, que por sua vez é constituído pelos atributos rua (VARCHAR(45)), localidade (VARCHAR(45)), código postal (VARCHAR(45)) e a chave primária id (INT) não nula, todos simples e só com um valor.

- **Relacionamento Beneficiário-Alimento**

O relacionamento entre o beneficiário e o alimento dá origem a três atributos simples e que têm só um valor, data (DATE), quantidade (INT) não nula e a chave primária id (INT) não nula.

- **Relacionamento Alimento-Recolha**

O relacionamento entre o beneficiário e o alimento dá origem ao atributo simples quantidade (INT) não nulo, que só tem um valor; e a chave primária idRecolha/Alimento (INT) não nula.

3.5 Apresentação e explicação do diagrama ER

Resumindo o nosso Modelo Conceptual, o Centro de Operações é constituído pela Fonte, Voluntário, Beneficiário e armazena Alimento. O Beneficiário recolhe Alimento, que por sua vez, constitui uma Recolha. A Fonte prepara a Recolha e o Voluntário fá-la.

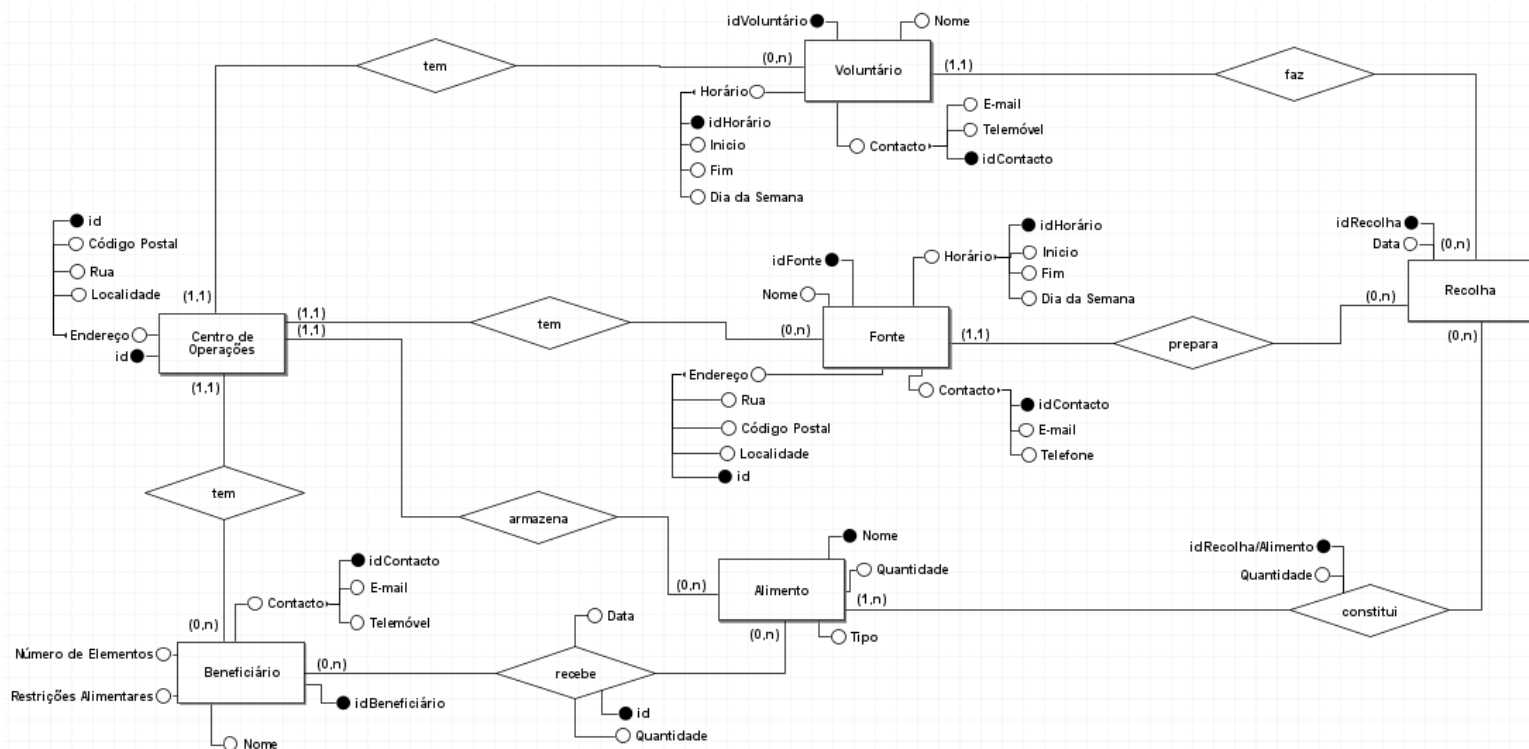


Figura 9 - Modelo conceptual desenvolvido

3.6 Validação do modelo de dados produzido

Como se trata de uma base de dados para um centro de voluntariado que envolve inúmeras transferências de alimentos, é importante que se consiga registar para além das entidades envolvidas nessas transações as informações das quantidades e tipos de alimentos envolvidos em cada uma delas.

Em relação a este objetivo é importante saber onde ir buscar os alimentos, quando, quem e exatamente qual o conteúdo dessa carga. É preciso depois registar o armazenamento desses alimentos no armazém do Centro de Operações e posteriormente indicar todas as retiradas desses mesmos alimentos por parte das famílias beneficiadas.

As Entidades responsáveis por carregar Alimentos são o Beneficiário no Centro de Operações e o Voluntário numa respetiva Fonte sendo estas bem representadas com os seus respetivos contactos e no caso de Fonte os respetivos Endereços para a criação fácil de rotas de movimentação.

Para sincronizar a data das recolhas ambos o Voluntário e Fonte têm um horário de funcionamento disponível.

Para registar o fluxo de cada alimento nas várias Recolhas e Entregas que irão ser efetuadas, é necessário indicar a quantidade e tipo de cada Alimento referente aquela transação numa certa data tal como se pode verificar nas relações entre Recolha/Alimento e Beneficiário/Alimento. A quantidade e tipo de alimentos presente no armazém também é representada neste modelo lógico como relação entre Centro de Operações e Alimento com a quantidade sendo atributo de Alimento.

Concluindo, o modelo conceptual mostra-se útil para responder aos requisitos anteriormente levantados.

4 Modelação Lógica

4.1 Construção e validação do modelo de dados lógico

A construção do modelo conceptual permitiu uma fácil identificação das entidades e atributos a serem colocados nas tabelas para construção do modelo lógico, seguindo as regras de derivação dos relacionamentos já impostas no modelo conceptual.

4.1.1 Relacionamentos de 1 para N

Para tratar estes relacionamentos, são necessárias duas entidades lógicas, uma para cada entidade, sendo a chave primária da entidade do lado 1 usada como chave estrangeira na entidade do lado N.

Ocorrências:

- “tem” entre Centro_de_operacoes(1) e Voluntário(N)
- “tem” entre Centro_de_operacoes(1) e Fonte(N)
- “armazena” entre Centro_de_operacoes(1) e Alimento(N)
- “tem” entre Centro_de_operacoes(1) e Beneficiário(N)
- “prepara” entre Fonte(1) e Recolha(N)
- “faz” entre Voluntário(1) e Recolha(N)

4.1.2 Relacionamentos de N para N

De modo a evitar complicações na fase de normalização, resolvemos desde início usar três entidade lógicas para cada relacionamento do tipo, ou seja, uma entidade lógica para cada entidade e uma para o relacionamento.

A uma entidade lógica correspondente à entidade é atribuída uma chave primária que será guardada como chave estrangeira na entidade lógica correspondente ao relacionamento.

Ocorrências:

- “recebe” entre Beneficiário(N) e Alimento(N)
- “constitui” entre Alimento(N) e Recolha(N)

Entidades resultantes

- **“Alimento/Beneficiário”** = {id, Quantidade, Data, Beneficiario_idBeneficiario, Alimento_Nome}
- **“Beneficiário”** = {idBeneficiario, Nome, Restricoes_Alimentares, Contacto_idContacto, Centro_de_Operacoes}
- **“Contacto”** = {idContacto, Email, Telefone}
- **“Endereco”** = {id, Rua, Codigo_postal, Localidade}
- **“Centro de Operacoes”** = {id, Endereco}
- **“Alimento”** = {Nome, Quantidade, Tipo, Centro_de_Operacoes}
- **“Recolha/Alimento”** = {idRecolha, Quantidade, Recolha_idRecolha, Alimento_Nome}
- **“Voluntario”** = {idVoluntario, Nome, Contacto_idContacto, Centro_de_Operacoes}
- **“Horario”** = {idHorario, Inicio, Fim, Dia_da_semana, Voluntario_idVoluntario, Fonte_idFonte}
- **“Fonte”** = {idFonte, Nome, Endereco_id, Centro_de_Operacoes}
- **“Recolha”** = {idRecolha, Data, Fonte_idFonte, Voluntario_idVoluntario}

4.2 Desenho do modelo lógico

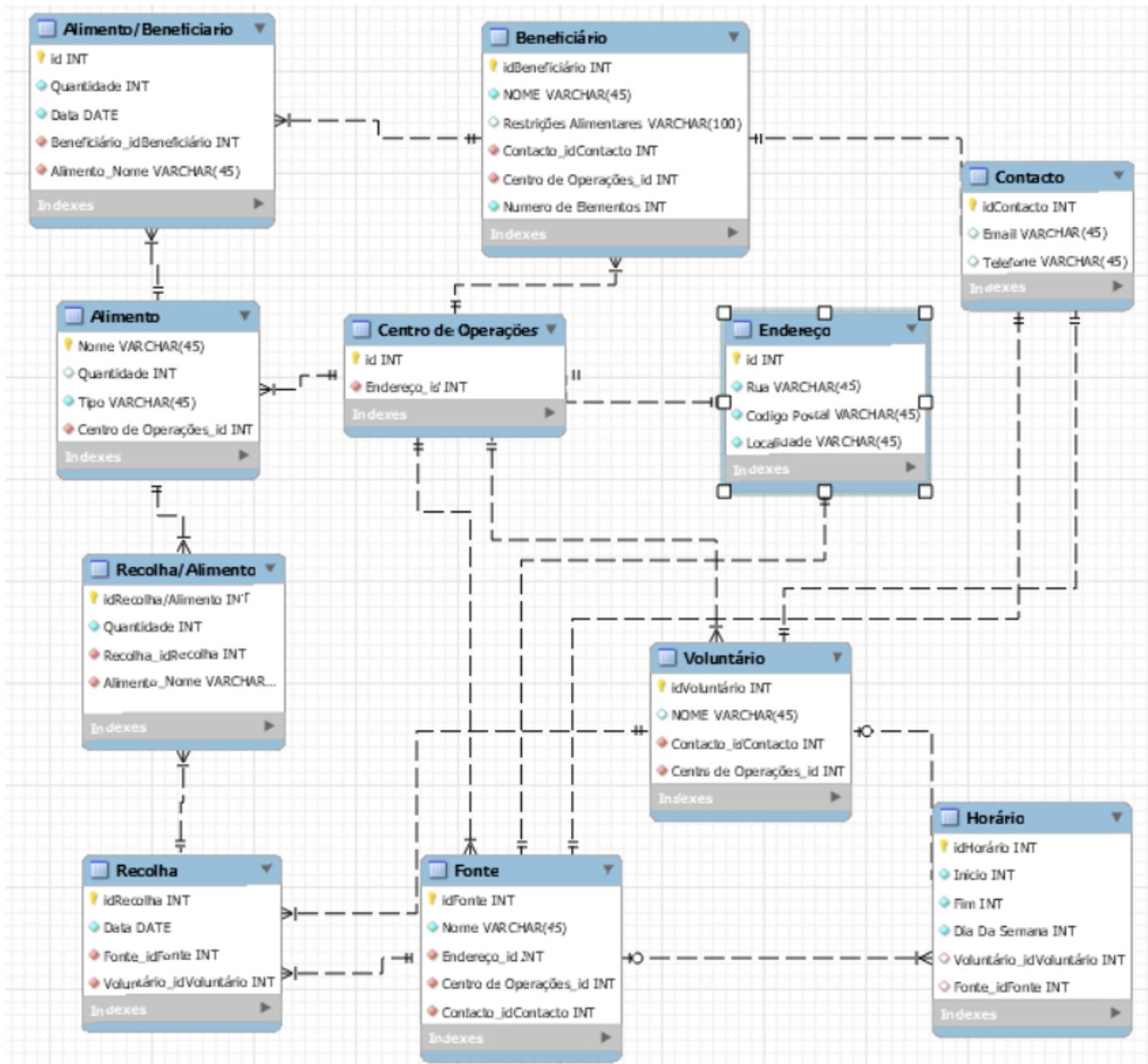


Figura 10 - Modelo lógico desenvolvido

4.3 Validação do modelo através da normalização

O modelo lógico verifica a primeira forma normal, uma vez que todas as entidades lógicas contêm uma chave primária, cada uma possui apenas atributos atômicos, não existindo atributos multivalorados ou compostos. Suponhamos que o voluntário não tinha um atributo identificador a servir de chave primária, neste caso, seria impossível haver distinção entre voluntários, pois o nome não o poderia ser, uma vez que podem existir voluntários com o mesmo nome.

O modelo lógico também verifica a segunda forma normal, uma vez que verifica a primeira forma normal e todos os atributos não chave são funcionalmente dependentes da chave primária.

Por fim, o modelo verifica a terceira forma normal, uma vez que verifica a segunda forma normal, e não existem dependências funcionais entre dois ou mais atributos não chave, ou seja, não existem atributos não chave a dependerem de um outro atributo não chave.

4.4 Validação do modelo com interrogações do utilizador

Para nos certificarmos que o modelo lógico se encontra válido, seleccionamos algumas interrogações relevantes que constam nos requisitos de exploração da base de dados e explicamos de que forma o modelo as consegue satisfazer.

1. O voluntário pode verificar a lista de todos os alimentos armazenados.

$$\pi_{\text{Nome, Quantidade}} (\sigma_{\text{Quantidade} \text{ DESC}} (\text{Alimento}))$$

Para efetuar a consulta começamos por ordenar os registos da tabela Alimento por ordem decrescente segundo a Quantidade, de seguida restringimos a tabela resultante a apenas duas colunas, Nome e Quantidade.

2. O beneficiário pode verificar a lista de entregas que recebeu, por data.

$$\pi_{\text{Data, Alimento, Quantidade}} (\sigma_{\text{Data} \text{ DESC}} ((\sigma_{\text{Nome} = k} (\text{Beneficiario})) \bowtie_{\text{idBeneficiario} = \text{Beneficiario}} (\text{Alimento/Beneficiario}))))$$

Inicialmente é feita uma seleção da tabela Beneficiário dos registos com o Nome “k”, de seguida uma junção da tabela resultante com a tabela Alimento/Beneficiario segundo o critério idBeneficiario = Beneficiario, de seguida é feita uma ordenação decrescente segundo a Data e por fim, restringimos a tabela resultante a apenas três colunas, Data, Alimento, Quantidade.

3. O Voluntário pode verificar a lista de recolhas que realizou, por data.

$$\pi_{\text{Data, idRecolha, Fonte}} (\sigma_{\text{Data} \text{ DESC}} ((\sigma_{\text{Nome} = k} ((\text{Fonte}) \bowtie_{\text{idFonte} = \text{Fonte}} ((\text{Voluntario}) \bowtie_{\text{idVoluntário} = \text{Voluntario}} (\text{Recolha}))))))$$

Inicialmente são feitas duas junções, uma da tabela Voluntário com a tabela Recolha segundo o critério idVoluntario = Voluntario, outra da tabela resultante desta com a tabela Fonte segundo o critério idFonte = Fonte. De seguida é feita uma seleção dos registos da tabela resultante das duas junções pelo Nome “k” e posteriormente uma

ordenação decrescente segundo a Data. Por fim restringimos a tabela resultante a apenas três colunas, Data, idRecolha, Fonte.

4. O Voluntário pode ver o horário de uma fonte.

$\Pi_{\text{Inicio, Fim, 'Dia da Semana'}} (\sigma_{\text{Nome} = k} ((\text{Fonte}) \bowtie_{\text{idFonte} = \text{Fonte}} (\text{Horario})))$

Inicialmente é feita a junção da tabela Fonte com a tabela Horario segundo o critério idFonte = Fonte, de seguida é feita uma seleção dos registos da tabela resultante pelo Nome “k” e por fim restringimos a tabela resultante a apenas três colunas, Inicio, Fim e Dia da Semana.

5. O Voluntário pode ver todos os alimentos e quantidade recolhidos numa data.

$\Pi_{\text{Alimento, Quantidade}} (\sigma_{\text{Data} = k} ((\text{Recolha/Alimento}) \bowtie_{\text{Recolha} = \text{idRecolha}} (\text{Recolha})))$

Inicialmente é feita a junção da tabela Recolha/Alimento com a tabela Recolha segundo o critério Recolha = idRecolha, de seguida é feita uma seleção dos registos da tabela resultante pela Data “k” e por fim restringimos a tabela resultante a apenas duas colunas, Alimento e Quantidade.

4.5 Revisão do modelo lógico produzido

O principal objetivo da produção do modelo lógico consiste em transformar o modelo conceptual em algo de mais baixo nível. Por outras palavras, em algo mais facilmente implementável numa máquina.

Como podemos verificar ao longo do capítulo 4, utilizando todas as entidades e relacionamentos construídos, o nosso modelo cumpre aquilo que lhe é pedido e por isso podemos afirmar que é uma representação correta para aquilo que foi pedido.

5 Implementação Física

5.1 Seleção do sistema de gestão de base de dados

O sistema de gestão de base de dados escolhido para implementar a base de dados para o Centro de voluntariado ReFood foi o MySQL.

O MySQL é conhecido por ser fácil de usar e, para além de ter uma grande base de utilizadores, tem um ótimo desempenho em eficiência, velocidade e segurança. É também o sistema aprendido nas aulas, e, conseqüentemente, aquele que este grupo está mais familiarizado.

5.2 Tradução do sistema lógico para sistema de gestão de bases de dados escolhido em SQL

Usando a ferramenta de administração MySQL Workbench que o MySQL oferece, é possível num processo automático, nomeado “Forward Engineering”, originar a partir do modelo lógico de alto nível, o código resultante, e posteriormente simplesmente executar a base de dados.

No entanto, antes de tal ter sido feito, foi realizada a implementação manualmente, utilizando os conceitos dados nas aulas e comparando-o ao automático, concluindo a boa realização da base de dados de ambas as formas.

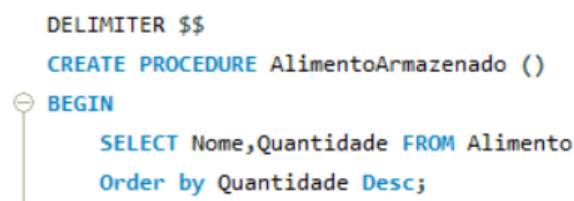
A base também foi povoada com vários dados para verificar a realização das *queries*.

5.3 Tradução das interrogações do utilizador para SQL

Para a implementação das interrogações, foram utilizados *stored procedures* que seguissem as metodologias previamente calculadas em álgebra relacional. Estes *stored procedures* ficam armazenados na base de dados para posterior rápido uso.

5.3.1 Alguns Exemplos

1. O Voluntário pode verificar a lista de todos os alimentos armazenados.

A screenshot of the MySQL Workbench interface showing a SQL editor. The text in the editor is: DELIMITER \$\$, CREATE PROCEDURE AlimentoArmazenado (), BEGIN, SELECT Nome,Quantidade FROM Alimento, Order by Quantidade Desc;. The text is color-coded: DELIMITER is blue, \$\$ is black, CREATE PROCEDURE is blue, AlimentoArmazenado is blue, () is black, BEGIN is blue, SELECT is blue, Nome,Quantidade is black, FROM is blue, Alimento is black, Order by is blue, Quantidade is black, Desc; is black.

```
DELIMITER $$
CREATE PROCEDURE AlimentoArmazenado ()
BEGIN
    SELECT Nome,Quantidade FROM Alimento
    Order by Quantidade Desc;
```

Figura 11 - Código obtido na Query 1

2. O Beneficiário pode verificar a lista de entregas que recebeu, por data.

```
DELIMITER $$
CREATE PROCEDURE BeneficiárioEntregas (IN nome VarChar(45))
BEGIN
    SELECT Data,Alimento,Quantidade FROM `Alimento/Beneficiário`
    Join Beneficiário ON idBeneficiário=`Alimento/Beneficiário`.Beneficiário
    where Beneficiário.Nome=nome
    order by Data Desc;
END $$
CALL BeneficiárioEntregas("Joel");
```

Figura 12 - Código obtido na Query 2

3. O Voluntário pode verificar a lista de recolhas que realizou, por data.

```
DELIMITER $$
CREATE PROCEDURE VoluntárioRecolhas (IN nome VarChar(45))
BEGIN
    SELECT IdRecolha,Data,Fonete.Nome FROM Recolha
    Join Voluntário ON IdVoluntário=Recolha.Voluntário
    Join Fonte ON idFonte=Recolha.Fonte
    where Voluntário.Nome=nome
    order by Data Desc;
END $$
CALL VoluntárioRecolhas("Sara Dias");
```

Figura 13 - Código obtido na Query 3

4. O Voluntário pode ver o horário de uma fonte.

```
-- DROP PROCEDURE HorárioFonte;
DELIMITER $$
CREATE PROCEDURE HorárioFonte (IN nome VarChar(45))
BEGIN
    SELECT Inicio,Fim,`Dia Da Semana` FROM Horário
    Join Fonte ON idFonte=Horário.Fonte
    Where Fonte.Nome=nome;
END $$
CALL HorárioFonte("PizzaHut");
```

Figura 14 - Código obtido na Query 4

5. O Voluntário pode ver todos os alimentos e quantidade recolhida numa data.

```
DELIMITER $$
CREATE PROCEDURE RecolhasData (IN data1 Date)
BEGIN
    SELECT Alimento,Quantidade FROM Recolha
    Join `Recolha/Alimento` On Recolha=idRecolha
    Where Data=data1;
END $$
```

Figura 15 - Código obtido na Query 5

5.4 Escolha, definição e caracterização de índices em SQL

Normalmente, todos os índices necessários são criados pelo MySQL durante o *CREATE TABLE statement*. Estes índices são maioritariamente as chaves primárias e estrangeiras de uma tabela e são necessários para a realização e operações sobre tabelas.

No entanto, é possível criar índices com o *statement CREATE INDEX*. Estes novos índices podem ajudar a aumentar o desempenho de certas *queries* que dependem dos valores que atribuímos ao índice. Por este motivo, consideramos necessário criar índices para estes nas tabelas de Recolha e Alimentos/Beneficiário, devido à quantidade de operações que se fazem sobre datas, como, por exemplo, saber como reunir recolhas numa certa data.

```
-- Criação de Indexes
-- Automatico para chaves estrangeiras, e chaves primárias
CREATE INDEX idx_DataRecolha on Recolha (Data);
CREATE INDEX idx_DataBeneficiário on `Alimento/Beneficiário` (Data);
```

Figura 16 - Índices adicionados

5.5 Triggers MySQL

Devido à forma como a nossa base de dados funciona, para podermos responder à *query* que indica que as quantidades de alimentos armazenados no Centro de Operações devem ser alteradas quando se realiza uma recolha ou entrega a um beneficiário, foi necessário usar um *trigger* que aumenta a quantidade do alimento sempre que se adiciona uma entrada na tabela Recolha/Alimento e diminuir a quantidade do alimento sempre que se adiciona uma entrada na tabela Alimento/Beneficiário.

```

-- TRIGGERS ATUALIZAR AS QUANTIDADES NO ARMAZEM EM RECOLHAS E ENTREGAS;
-- DROP TRIGGER Recolha;
DELIMITER $$
CREATE TRIGGER Recolha
    AFTER INSERT ON `Recolha/Alimento`
    FOR EACH ROW
BEGIN
    UPDATE Alimento SET Alimento.Quantidade = Alimento.Quantidade + new.Quantidade
        where Alimento.Nome=new.Alimento;
END $$

-- DROP TRIGGER Entrega;
DELIMITER $$
CREATE TRIGGER Entrega
    AFTER INSERT ON `Alimento/Beneficiário`
    FOR EACH ROW
BEGIN
    UPDATE Alimento SET Alimento.Quantidade = Alimento.Quantidade - new.Quantidade
        where Alimento.Nome=new.Alimento;
END $$

```

Figura 17 - Triggers criados

5.6 Estimativa do espaço em disco de base de dados e taxa de crescimento anual

Tabela	Espaço Esperado Por Entrada
Alimento	102 bytes
Alimento / Beneficiário	62 bytes
Beneficiário	165 bytes
Centro de Operações	8 bytes
Contacto	98 bytes
Endereço	145 bytes
Fonte	63 bytes
Horário	24 bytes
Recolha	15 bytes
Recolha/Alimento	59 bytes
Voluntário	59 bytes

Tabela 1 - Espaço em disco esperado por entrada

O projeto ReFood contém, na atualidade, considerando um Centro de Operações, aproximadamente umas 120 famílias (beneficiários), 50 fontes e 200 voluntários. Cada fonte oferece 1 recolha por dia, equivalente a 7 horários. Cada recolha tem em média 5 diferentes tipos de alimentos oferecidos e existem uns 50 diferentes alimentos armazenados no Centro de Operações, ao longo do ano. Assumindo que existem 50 famílias a receber alimentos todos os dias, podemos fazer os cálculos ao longo do ano.

- **Beneficiários:** $120 * 165 + 120 * 98 = 32 \text{ Kb}$
- **Fontes:** $50 * 63 + 50 * 145 + 50 * 98 + 50 * 7 * 24 = 23.7 \text{ Kb}$
- **Voluntários:** $200 * 59 + 200 * 98 = 31.4 \text{ Kb}$
- **Alimentos:** $50 * 102 = 5.1 \text{ Kb}$
- **Recolhas:** $365 * (50 * 15 + 50 * 5 * 59) = 5657.5 \text{ Kb}$
- **Alimento/Beneficiário:** $365 * 50 * 62 = 1131.5 \text{ Kb}$
- **Total:** 6.9 Mb ao final de um ano

5.7 Definição e caracterização das vistas de utilização em SQL

Para ajudar na realização de *queries* mais complexas, foram adicionadas 2 novas vistas. Uma delas tem o intuito de ajudar as operações que envolvem a gestão de alimentos no armazém do Centro de Operações ao longo do tempo. Para tal, esta nova tabela teria de registar o fluxo de alimentos dado pelas recolhas e entregas realizadas de cada alimento e as respetivas datas. A outra vista tem como objetivo ajudar na realização de rotas (endereços das fontes) e horários das fontes para que os voluntários possam obter o máximo de alimentos na menor distância percorrida possível.

```
-- DROP VIEW view_fonte_Endereço;
-- SELECT * FROM view_fonte_Endereço;
CREATE VIEW view_fonte_Endereço AS
SELECT Fonte.Nome,Rua,`Codigo-Postal`,Localidade,Inicio,Fim,`Dia Da Semana` From Fonte
JOIN Endereço ON Fonte.Endereço=Endereço.id
JOIN Horário ON Horário.Fonte=Fonte.idFonte
ORDER BY `Dia Da Semana`;

-- DROP VIEW fluxo_alimento;
-- SELECT * FROM fluxo_alimento;
CREATE VIEW fluxo_alimento AS
SELECT Alimento,Quantidade,Data FROM
    Recolha JOIN `Recolha/Alimento` ON `Recolha/Alimento`.Recolha=Recolha.idRecolha
    UNION SELECT Alimento,-Quantidade,Data FROM `Alimento/Beneficiário`
ORDER BY Data;
```

Figura 18 - Vistas adicionadas

5.8 Revisão Do Sistema Implementado

Para validar o nosso sistema, no final, foi reunido com o cliente de forma a explicar o funcionamento deste e verificar se todos os seus requisitos estavam cumpridos.

Numa primeira fase, foram adicionadas várias amostras à base de dados. Numa segunda fase, foram realizadas as *queries* (*Stored Procedures*) e verificado se estas continham toda a informação necessária e correta. Numa terceira fase, falamos do espaço ocupado previsto ao longo do tempo.

Numa fase final, foram pedidas, por eventuais modificações, novas *queries* que o cliente desejava implementadas, esclarecimento de dúvidas e permissão para continuar o projeto.

6 Conclusões

Com a realização deste trabalho, conseguimos aplicar os conhecimentos adquiridos durante as aulas teóricas e práticas de Bases de Dados, a um contexto da vida real e que é próximo a um dos membros do nosso grupo.

Durante a sua realização, como se trata de uma organização verdadeira, através dos nossos conhecimentos e da pesquisa feita, tentamos deixar o projeto o mais realista possível.

Para a ReFood, a criação desta base de dados ajudaria a tornar o processo de monitorização das atividades mais fácil e rápido, sendo muito mais fácil organizar as recolhas dos alimentos, as suas entregas aos beneficiários e incluir de forma dinâmica o voluntário e os alimentos no sistema.

Durante o trabalho, com o intuito de nunca realizar trabalho desnecessário ou fazer um desenvolvimento sem o devido planeamento, tivemos de respeitar os conceitos do ciclo de vida para o desenvolvimento da base de dados falados na aula, repartindo o processo em diversas partes metódicas. Começando com o estudo na vida real do fenómeno ReFood, partindo daí para a criação de vários requisitos que posteriormente iriam ditar o funcionamento da nossa base de dados. Todos os passos seguintes devem ter em conta esses requisitos, sendo que o seu cumprimento é o objetivo final. Seguidamente formamos um modelo conceptual para ter alguma ideia das entidades e relacionamentos de forma abstrata seguido de um modelo lógico, fazendo muito mais sentido computacionalmente daquilo que queremos criar. Por fim bastou implementar a base de dados em SQL adicionando a criação das *queries* que respondessem aos requisitos.

Algumas das dificuldades que enfrentamos no trabalho foi, por exemplo, como representar o atributo horário. No início, começou por ser um atributo simples e multivalorado, em que cada INT seria referente a um horário específico. No entanto, decidimos torná-lo num atributo composto por início, fim e o dia da semana.

Para além disso, considerávamos, no início, que as recolhas seriam várias relações entre as entidades. Mas, ao longo da realização do trabalho, achamos que não havia necessidade de estar a repetir constantemente a mesma relação com os mesmos atributos, e por isso, consideramos que seria mais fácil criar uma entidade que representasse uma recolha.

Para concluir, consideramos que conseguimos desenvolver uma boa base de dados, para o nosso caso real, apesar de que, alguns aspetos teriam de ser mais complexos caso isto se tratasse de uma base de dados no mundo real, como por exemplo, a inclusão de mais elementos na base de dados, entre eles, o desempenho de mais funções por parte dos voluntários.

Anexos

I. Anexo 1 – Script de Inicialização da Base de Dados

```
-- DROP SCHEMA Refood;
CREATE SCHEMA Refood;
USE Refood;

-- DROP TABLE Contacto;
CREATE TABLE Contacto (
    idContacto int NOT NULL,
    Email VARCHAR(45),
    Telefone VARCHAR(45),
    PRIMARY KEY (idContacto)
);

-- DROP TABLE Endereço;
CREATE TABLE Endereço (
    id int NOT NULL,
    Rua VARCHAR(45),
    `Codigo-Postal` VARCHAR(45),
    Localidade VARCHAR(45),
    PRIMARY KEY (id)
);

-- DROP TABLE `Centro de Operações`;
CREATE TABLE `Centro de Operações` (
    id INT NOT NULL,
    Endereço int NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT `fk_Centro_Endereço`
        FOREIGN KEY (Endereço)
        REFERENCES Endereço (id)
);

-- DROP TABLE Beneficiário;
CREATE TABLE Beneficiário (
    idBeneficiário int NOT NULL,
    Nome VARCHAR (45),
    `Restrições Alimentares` VARCHAR (100),
    `Numero de Elementos` int NOT NULL,
```

```

Contacto int,
`Centro de Operações` int NOT NULL,
PRIMARY KEY (idBeneficiário),
CONSTRAINT `fk_Beneficiario_Centro`
    FOREIGN KEY (`Centro de Operações`)
    REFERENCES `Centro de Operações` (id),
CONSTRAINT `fk_Beneficiario_Contacto`
    FOREIGN KEY (Contacto)
    REFERENCES Contacto (idContacto)
);

```

```

-- DROP TABLE Voluntário;
CREATE TABLE Voluntário (
idVoluntário int NOT NULL,
Nome VARCHAR(45),
Contacto int,
`Centro de Operações` int NOT NULL,
PRIMARY KEY (idVoluntário),
CONSTRAINT `fk_Voluntário_Centro`
    FOREIGN KEY (`Centro de Operações`)
    REFERENCES `Centro de Operações` (id),
CONSTRAINT `fk_Voluntário_Contacto`
    FOREIGN KEY (Contacto)
    REFERENCES Contacto (idContacto)
);

```

```

-- DROP TABLE Fonte;
CREATE TABLE Fonte (
idFonte int NOT NULL,
Nome VARCHAR(45),
Endereço int NOT NULL,
`Centro de Operações` int NOT NULL,
Contacto int NOT NULL,
PRIMARY KEY (idFonte),
CONSTRAINT `fk_Fonte_Endereço`
    FOREIGN KEY (Endereço)
    REFERENCES Endereço (id),
CONSTRAINT `fk_Fonte_Centro`
    FOREIGN KEY (`Centro de Operações`)
    REFERENCES `Centro de Operações` (id),
CONSTRAINT `fk_Fonte_Contacto`

```

```

        FOREIGN KEY (`Contacto`)
        REFERENCES `Contacto` (idContacto)
    );

-- DROP TABLE Alimento;
CREATE TABLE Alimento (
    Nome VARCHAR(45) NOT NULL,
    Quantidade int NOT NULL,
    Tipo VARCHAR(45),
    `Centro de Operações` int NOT NULL,
    PRIMARY KEY (Nome),
    CONSTRAINT `fk_Alimento_Centro`
        FOREIGN KEY (`Centro de Operações`)
        REFERENCES `Centro de Operações` (id)
);

-- DROP TABLE Recolha;
CREATE TABLE Recolha (
    idRecolha int NOT NULL,
    Data Date,
    Fonte int NOT NULL,
    Voluntário int NOT NULL,
    PRIMARY KEY (idRecolha),
    CONSTRAINT `fk_Recolha_Fonte`
        FOREIGN KEY (Fonte)
        REFERENCES Fonte (idFonte),
    CONSTRAINT `fk_Recolha_Voluntário`
        FOREIGN KEY (Voluntário)
        REFERENCES Voluntário (idVoluntário)
);

-- DROP TABLE `Recolha/Alimento`;
CREATE TABLE `Recolha/Alimento` (
    `idRecolha/Alimento` int NOT NULL,
    Quantidade int NOT NULL,
    Recolha int NOT NULL,
    Alimento VARCHAR(45) NOT NULL,
    PRIMARY KEY(`idRecolha/Alimento`),
    CONSTRAINT `fk_RA/Alimento`
        FOREIGN KEY (Alimento)
        REFERENCES Alimento (Nome),

```

```

CONSTRAINT `fk_RA/Recolha`
    FOREIGN KEY (Recolha)
    REFERENCES Recolha (idRecolha)
);

-- DROP TABLE `Alimento/Beneficiário`;
CREATE TABLE `Alimento/Beneficiário` (
    id int NOT NULL,
    Quantidade int NOT NULL,
    Data Date,
    Beneficiário int NOT NULL,
    Alimento VARCHAR(45) NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT `fk_AB/Alimento`
        FOREIGN KEY (Alimento)
        REFERENCES Alimento (Nome),
    CONSTRAINT `fk_AB/Beneficiario`
        FOREIGN KEY (Beneficiário)
        REFERENCES Beneficiário (idBeneficiário)
);

-- DROP TABLE Horário;
CREATE TABLE Horário (
    idHorário int NOT NULL,
    Início int NOT NULL,
    Fim int NOT NULL,
    `Dia Da Semana` int NOT NULL,
    Voluntário int,
    Fonte int,
    PRIMARY KEY (idHorário),
    CONSTRAINT `fk_Horario/Voluntário`
        FOREIGN KEY (Voluntário)
        REFERENCES Voluntário (idVoluntário),
    CONSTRAINT `fk_Horario/Fonte`
        FOREIGN KEY (Fonte)
        REFERENCES Fonte (idFonte)
);

```

II. Anexo 2 – Script de População Inicial

```
-- Popular Amostra

-- Popular Contactos
-- SELECT * FROM Contacto;

INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(1,"jose@hotmail.com","989767565");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(2,"a@hotmail.com","989767566");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(3,"b@hotmail.com","989767567");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(4,"c@hotmail.com","989767568");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(5,"d@hotmail.com","989767569");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(6,"e@hotmail.com","989767510");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(7,"f@hotmail.com","989767511");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(8,"g@hotmail.com","989767512");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(9,"h@hotmail.com","989767513");
INSERT      Into      Contacto      (idContacto,Email,Telefone)      VALUES
(10,"i@hotmail.com","989767514");

-- Popular Endereços
-- SELECT * FROM Endereço;

INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (1,"Rua
25 de Abril","4000-700","Lisboa");
INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (2,"Rua
26 de Abril","4000-701","Lisboa");
INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (3,"Rua
27 de Abril","4000-702","Lisboa");
INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (4,"Rua
28 de Abril","4000-703","Lisboa");
INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (5,"Rua
29 de Abril","4000-704","Lisboa");
```

```

INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (6,"Rua
30 de Abril","4000-705","Lisboa");
INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (7,"Rua
31 de Abril","4000-706","Lisboa");
INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (8,"Rua
32 de Abril","4000-707","Lisboa");
INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (9,"Rua
33 de Abril","4000-708","Lisboa");
INSERT Into Endereço (id,Rua,`Codigo-Postal`,Localidade) VALUES (10,"Rua
25 de Dezembro","4000-709","Lisboa");

-- Popular Centro de Observações
-- SELECT * FROM `Centro de Operações`;
INSERT Into `Centro de Operações` (id,Endereço) VALUES (1,1);

-- Popular Voluntários
-- SELECT * FROM Voluntário;
INSERT Into Voluntário (idVoluntário,Nome,Contacto,`Centro de
Operações`) VALUES (1,"Sara Dias",1,1);
INSERT Into Voluntário (idVoluntário,Nome,Contacto,`Centro de
Operações`) VALUES (2,"Sara1 Dias",2,1);
INSERT Into Voluntário (idVoluntário,Nome,Contacto,`Centro de
Operações`) VALUES (3,"Sara2 Dias",3,1);

-- Popular Beneficiários
-- SELECT * FROM Beneficiário;
INSERT Into Beneficiário (idBeneficiário,Nome,`Restrições
Alimentares`,Contacto,`Numero de Elementos`,`Centro de Operações`)
VALUES (1,"Joel",NULL,4,10,1);
INSERT Into Beneficiário (idBeneficiário,Nome,`Restrições
Alimentares`,Contacto,`Numero de Elementos`,`Centro de Operações`)
VALUES (2,"Joel2","Lacticios",5,11,1);
INSERT Into Beneficiário (idBeneficiário,Nome,`Restrições
Alimentares`,Contacto,`Numero de Elementos`,`Centro de Operações`)
VALUES (3,"Joel3",NULL,6,11,1);

-- Popular Fontes
-- SELECT * FROM Fonte;
INSERT Into Fonte (idFonte,Nome,Endereço,`Centro de Operações`,Contacto)
VALUES (1,"PizzaHut",2,1,7);

```

```

INSERT Into Fonte (idFonte, Nome, Endereço, `Centro de Operações`, Contacto)
VALUES (2, "PizzaHut2", 3, 1, 8);
INSERT Into Fonte (idFonte, Nome, Endereço, `Centro de Operações`, Contacto)
VALUES (3, "PizzaHut3", 4, 1, 9);

-- Popular Alimentos
-- SELECT * FROM Alimento;
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Cenoura", 100, "Vegetal", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Massa", 5, "Processada", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Tomate", 6, "Fruto", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Arroz", 100, "Processada", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Chocolate", 10, "Doce", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Leite", 7, "Bebida", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Pizza", 100, "Cozinhada", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Sopa", 100, "Por Cozinhar", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Cogumelo", 9, "Vegetal", 1);
INSERT Into Alimento (Nome, Quantidade, Tipo, `Centro de Operações`) VALUES
("Banana", 100, "Fruto", 1);

-- Popular Horários
-- SELECT * FROM Horário;
INSERT      Into      Horário      (idHorário, Inicio, Fim, `Dia      Da
Semana`, Voluntário, Fonte) VALUES (1, 8, 10, 1, 1, null);
INSERT      Into      Horário      (idHorário, Inicio, Fim, `Dia      Da
Semana`, Voluntário, Fonte) VALUES (2, 9, 10, 2, 2, null);
INSERT      Into      Horário      (idHorário, Inicio, Fim, `Dia      Da
Semana`, Voluntário, Fonte) VALUES (3, 8, 11, 3, 3, null);
INSERT      Into      Horário      (idHorário, Inicio, Fim, `Dia      Da
Semana`, Voluntário, Fonte) VALUES (4, 10, 12, 4, null, 1);
INSERT      Into      Horário      (idHorário, Inicio, Fim, `Dia      Da
Semana`, Voluntário, Fonte) VALUES (5, 10, 15, 5, null, 2);

```



```

INSERT      Into      Horário      (idHorário,Inicio,Fim,`Dia      Da
Semana`,Voluntário,Fonte) VALUES (6,16,18,6,null,3);
INSERT      Into      Horário      (idHorário,Inicio,Fim,`Dia      Da
Semana`,Voluntário,Fonte) VALUES (7,19,20,7,null,1);

-- Popular Recolhas
-- SELECT * FROM Recolha;
INSERT Into Recolha (idRecolha,Data,Fonte,Voluntário) VALUES (1,'2010-
11-30',1,1);
INSERT Into Recolha (idRecolha,Data,Fonte,Voluntário) VALUES (2,'2010-
11-29',2,1);
INSERT Into Recolha (idRecolha,Data,Fonte,Voluntário) VALUES (3,'2010-
11-28',3,1);
INSERT Into Recolha (idRecolha,Data,Fonte,Voluntário) VALUES (4,'2010-
11-27',1,2);
INSERT Into Recolha (idRecolha,Data,Fonte,Voluntário) VALUES (5,'2010-
11-27',1,2);
INSERT Into Recolha (idRecolha,Data,Fonte,Voluntário) VALUES (6,'2010-
11-26',1,2);

-- Popular Recolha/Alimento
-- SELECT * FROM `Recolha/Alimento`;
INSERT      Into      `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)      VALUES
(1,10,1,"Cenoura");
INSERT      Into      `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)      VALUES
(2,5,1,"Massa");
INSERT      Into      `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)      VALUES
(3,6,2,"Cenoura");
INSERT      Into      `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)      VALUES
(4,7,3,"Pizza");
INSERT      Into      `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)      VALUES
(5,8,4,"Leite");
INSERT      Into      `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)      VALUES
(6,9,5,"Sopa");

```

```

INSERT                                Into                                `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)                VALUES
(7,10,6,"Banana");

INSERT                                Into                                `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)                VALUES
(8,11,3,"Cogumelo");

INSERT                                Into                                `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)                VALUES
(9,12,3,"Tomate");

INSERT                                Into                                `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)                VALUES
(10,13,3,"Chocolate");

-- Popular Alimento/Beneficiário
-- SELECT * FROM `Alimento/Beneficiário`:

INSERT                                Into                                `Alimento/Beneficiário`
(id,Quantidade,Data,Beneficiário,Alimento)                VALUES      (1,5,'2020-11-
27',1,"Cenoura");

INSERT                                Into                                `Alimento/Beneficiário`
(id,Quantidade,Data,Beneficiário,Alimento)                VALUES      (2,10,'2020-11-
28',1,"Massa");

INSERT                                Into                                `Alimento/Beneficiário`
(id,Quantidade,Data,Beneficiário,Alimento)                VALUES      (3,20,'2020-11-
29',2,"Pizza");

INSERT                                Into                                `Alimento/Beneficiário`
(id,Quantidade,Data,Beneficiário,Alimento)                VALUES      (4,25,'2020-11-
30',2,"Chocolate");

INSERT                                Into                                `Alimento/Beneficiário`
(id,Quantidade,Data,Beneficiário,Alimento)                VALUES      (5,30,'2020-01-
03',3,"Leite");

INSERT                                Into                                `Alimento/Beneficiário`
(id,Quantidade,Data,Beneficiário,Alimento)                VALUES      (6,35,'2020-02-
04',3,"Tomate");

```

III. Anexo 3 – Scripts das queries

```
-- Queries

-- O Voluntário pode verificar a lista de todos os alimentos armazenados
-- DROP PROCEDURE AlimentoArmazenado;
DELIMITER $$
CREATE PROCEDURE AlimentoArmazenado ()
BEGIN
    SELECT Nome,Quantidade FROM Alimento
    Order by Quantidade Desc;
END $$
-- CALL AlimentoArmazenado();

-- O Beneficiário pode verificar a lista de entregas que recebeu, por
data.
-- DROP PROCEDURE BeneficiárioEntregas;
DELIMITER $$
CREATE PROCEDURE BeneficiárioEntregas (IN nome VarChar(45))
BEGIN
    SELECT Data,Alimento,Quantidade FROM `Alimento/Beneficiário`
    Join                               Beneficiário                      ON
idBeneficiário=`Alimento/Beneficiário`.Beneficiário
    where Beneficiário.Nome=nome
    order by Data Desc;
END $$
-- CALL BeneficiárioEntregas("Joel");

-- O Voluntário pode verificar a lista de recolhas que realizou, por
data.
-- DROP PROCEDURE BeneficiárioEntregas;
DELIMITER $$
CREATE PROCEDURE VoluntárioRecolhas (IN nome VarChar(45))
BEGIN
    SELECT IdRecolha,Data,Fonte.Nome FROM Recolha
    Join Voluntário ON IdVoluntário=Recolha.Voluntário
    Join Fonte ON idFonte=Recolha.Fonte
    where Voluntário.Nome=nome
    order by Data Desc;
END $$
```

```

-- CALL VoluntárioRecolhas("Sara Dias");

-- O Voluntário pode ver o horário de uma fonte.
-- DROP PROCEDURE HorárioFonte;
DELIMITER $$
CREATE PROCEDURE HorárioFonte (IN nome VarChar(45))
BEGIN
    SELECT Inicio,Fim,`Dia Da Semana` FROM Horário
    Join Fonte ON idFonte=Horário.Fonte
    Where Fonte.Nome=nome;
END $$
CALL HorárioFonte("PizzaHut");

-- O Voluntário pode ver todos os alimentos e quantidade recolhidos numa
data.
-- DROP PROCEDURE RecolhasData;
DELIMITER $$
CREATE PROCEDURE RecolhasData (IN data1 Date)
BEGIN
    SELECT Alimento,Quantidade FROM Recolha
    Join `Recolha/Alimento` On Recolha=idRecolha
    Where Data=data1;
END $$
-- CALL RecolhasData('2010-11-27');

```

IV. Anexo 4 – Scripts das Views, Triggers e Indexes

```
-- Criação de Indexes
-- Automatico para chaves estrangeiras, e chaves primárias
CREATE INDEX idx_DataRecolha on Recolha (Data);
CREATE INDEX idx_DataBeneficiário on `Alimento/Beneficiário` (Data);

-- Criação de Views

-- DROP VIEW view_fonte_Endereço;
-- SELECT * FROM view_fonte_Endereço;
CREATE VIEW view_fonte_Endereço AS
SELECT Fonte.Nome,Rua,`Codigo-Postal`,Localidade,Inicio,Fim,`Dia Da
Semana` From Fonte
JOIN Endereço ON Fonte.Endereço=Endereço.id
JOIN Horário ON Horário.Fonte=Fonte.idFonte
ORDER BY `Dia Da Semana`;

-- DROP VIEW fluxo_alimento;
-- SELECT * FROM fluxo_alimento;
CREATE VIEW fluxo_alimento AS
SELECT Alimento,Quantidade,Data FROM
    Recolha JOIN `Recolha/Alimento` ON
`Recolha/Alimento`.Recolha=Recolha.idRecolha
    UNION SELECT Alimento,-Quantidade,Data FROM `Alimento/Beneficiário`
ORDER BY Data;

-- TRIGGERS ATUALIZAR AS QUANTIDADES NO ARMAZEM EM RECOLHAS E ENTREGAS;
-- DROP TRIGGER Recolha;
DELIMITER $$
CREATE TRIGGER Recolha
    AFTER INSERT ON `Recolha/Alimento`
    FOR EACH ROW
BEGIN
    UPDATE Alimento SET Alimento.Quantidade = Alimento.Quantidade +
new.Quantidade
        where Alimento.Nome=new.Alimento;
END $$
```

```

-- DROP TRIGGER Entrega;
DELIMITER $$
CREATE TRIGGER Entrega
    AFTER INSERT ON `Alimento/Beneficiário`
    FOR EACH ROW
BEGIN
    UPDATE Alimento SET Alimento.Quantidade = Alimento.Quantidade -
new.Quantidade
        where Alimento.Nome=new.Alimento;
END $$

/*
-- Teste dos Triggerts
SELECT Nome,Quantidade FROM Alimento
    Order by Quantidade Desc;
INSERT                Into                `Recolha/Alimento`
(`idRecolha/Alimento`,Quantidade,Recolha,Alimento)                VALUES
(12,200,1,"Chocolate");
INSERT                INTO                `Alimento/Beneficiário`
(id,Quantidade,Data,Beneficiário,Alimento)                VALUES    (7,100,'2020-11-
27',1,"Chocolate");
*/

```