



Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Letivo de 2018/2019

Gestão de uma Unidade Hospitalar

Carolina Cunha, A80142

João Pimentel, A80874

Pedro Gonçalves, A82313

Rodolfo Silva, A81716

Janeiro 2019

B
D

Data de Receção	
Responsável	
Avaliação	
Observações	

Gestão de uma Unidade Hospitalar

Carolina Cunha, A80142

João Pimentel, A80874

Pedro Gonçalves, A82313

Rodolfo Silva, A81716

Janeiro 2019

Resumo

Uma Base de Dados consiste numa coleção compartilhada de dados relacionados logicamente e na descrição dos mesmos, sendo o objetivo satisfazer as necessidades de uma organização [3]. As BD apresentam vantagens comparativamente aos sistemas de gestão de dados baseados em ficheiros, porque são mais eficientes, tanto na gestão como na procura de informação e segurança dos dados. Os SGBD são aplicações informáticas desenvolvidas para armazenar e gerir BD, onde o utilizador consegue definir, criar, manter e controlar o acesso aos dados.

O objetivo deste projeto é o desenvolvimento de um módulo de gestão de uma pequena Unidade Hospitalar. Para tal, foi criado um modelo que, para além de permitir agendar e realizar consultas, possibilita o acesso ao histórico médico de cada utente e o acesso ao histórico de consultas realizadas na instituição, permitindo assim uma maior gestão, tanto de recursos, como de tempo.

O projeto desenvolvido proporciona o fácil acesso a diferentes tipos de dados da Unidade Hospitalar, através de um modelo de utilização intuitiva por parte dos funcionários da mesma.

Área de Aplicação: Projeção de um Sistema de Base de Dados para uma Unidade Hospitalar que permita a gestão de todas as atividades realizadas por esse mesmo Centro de Saúde.

Palavras-Chave: Bases de Dados Relacionais, SQL, MySQL Workbench.

Índice

1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	1
1.3. Motivação e Objetivos	1
2. Levantamento e Análise de Requisitos	3
2.1. Método de levantamento e de análise de requisitos adotado	3
2.2. Requisitos levantados	3
2.2.1. Requisitos de descrição	3
2.2.2. Requisitos de exploração	4
2.2.3 Requisitos de controlo	4
2.3. Análise geral dos requisitos	4
3. Modelação Conceptual	6
3.1. Apresentação da abordagem de modelação realizada	6
3.2. Identificação e caracterização das entidades	6
3.3. Identificação e caracterização dos relacionamentos	7
3.4. Identificação e caracterização dos Atributos com as Entidades e Relacionamentos	7
3.5. Apresentação e explicação do diagrama ER	9
3.6. Validação do modelo de dados com o utilizador	10
4. Modelação Lógica	11
4.1. Construção do modelo de dados lógico	12
4.1.1. Tabela Utente	12
4.1.2. Tabela Incapacidade	12
4.1.3. Tabela Utente_Incapacitado	12
4.1.4. Tabela Contacto_Utente	13
4.1.5. Tabela PUC	13
4.1.6. Tabela Profissional_Saude	13
4.1.7. Tabela Especialidade	14
4.1.8. Tabela Contacto_PS	14
4.1.9. Tabela Consulta	14

4.1.10. Tabela Medicamento	15
4.1.11. Tabela Receita	15
4.1.12. Tabela Medicamento_Receita	15
4.2. Validação do Modelo através da Normalização	16
4.3. Validação do Modelo com as interrogações do Utilizador	16
4.4. Validação do modelo com as transações estabelecidas	17
4.5. Revisão do modelo lógico com o utilizador	18
5. Implementação Física	19
5.1. Seleção do sistema de gestão de bases de dados	19
5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL	19
5.2.1. Tabela Utente	20
5.2.2. Tabela Contacto_Utente	21
5.2.3. Tabela Incapacidade	22
5.2.4. Tabela Utente_Incapacitado	22
5.2.5. Tabela Profissional_Saude	23
5.2.6. Tabela Contacto_PS	24
5.2.7. Tabela Especialidade	24
5.2.8. Tabela Consulta	25
5.2.9. Tabela PUC	26
5.2.10. Tabela Receita	27
5.2.11. Tabela Medicamento	27
5.2.12. Tabela Medicamento_Receita	28
5.3. Tradução das interrogações do utilizador para SQL (alguns exemplos)	29
5.3.1. Funções	29
5.3.2. Procedimentos	29
5.4. Tradução das transações estabelecidas para SQL (alguns exemplos)	30
5.4.1. Marcação de uma Consulta	30
5.4.2. Registo de um PS	31
5.4.3. Remarcar uma Consulta	31
5.5. Escolha, definição e caracterização de índices em SQL	32
5.6. Estimativa do espaço em disco da base de dados e taxa de crescimento anual	32

5.7. Definição e caracterização das vistas de utilização em SQL	34
5.7.1. Vista Medicamento	34
5.7.2. Vista Utente	35
5.7.3. Vista óbito	35
5.8. Definição e caracterização dos mecanismos de segurança em SQL	35
5.9. Revisão do sistema implementado com o utilizador	36
6. NoSQL	37
6.1. Justificação da utilização de um sistema NoSQL	37
6.2. Identificação e descrição dos objetivos da base de dados	38
6.3. Identificação e explicação do tipo de questões que serão realizadas sobre o sistema de dados NoSQL	38
6.4. Estrutura base para o sistema de dados NoSQL	39
6.5. Identificação dos objetos de dados no sistema SQL que serão utilizados para alimentar o novo sistema	40
6.6. Processo de migração de dados	40
6.7. Explicação do processo de migração de dados	40
6.8. Implementação do processo de migração de dados	41
6.9. Apresentar a forma como as questões identificadas previamente podem ser satisfeitas com o novo sistema	43
7. Conclusões e Trabalho Futuro	44
8. Referências	45
Lista de Siglas e Acrónimos	46
Anexos	47
Anexo I – Código DDL	47
Anexo II – Código do Povoamento Inicial	52
Anexo III – Tabelas Povoadas	58
Anexo IV – Código Relativo a Interrogações e Transações	62
Anexo V – Script de migração dos dados de MySQL para MongoDB	74

Índice de Figuras

Figura 1 - Modelo Conceptual da BD	9
Figura 2 - Modelo Lógico da BD	11
Figura 3 - Conteúdo da tabela Consulta no Modelo Físico	19
Figura 4 - Tabelas do Modelo Físico	19
Figura 5 - Tabela que representa a entidade Utente.....	20
Figura 6 - Trigger AtualizarFaixaEtaria.....	20
Figura 7 - Triggers GenderSafe e GenderSafePS.....	21
Figura 8 - Código de criação tabela Utente	21
Figura 9 - Tabela que representa a entidade Contacto_Utente	21
Figura 10 - Código de criação tabela Contacto_Utente.....	22
Figura 11 - Tabela que representa a entidade Incapacidade	22
Figura 12 - Código de criação tabela Incapacidade	22
Figura 13 - Tabela que representa a entidade Utente_Incapacitado	22
Figura 14 - Código de criação tabela Utente_Incapacitado.....	23
Figura 15 - Tabela que representa a entidade Profissional_Saude	23
Figura 16 - Código de criação tabela Profissional_Saude.....	23
Figura 17 - Tabela que representa a entidade Contacto_PS	24
Figura 18 - Código de criação tabela Contacto_PS	24
Figura 19 - Tabela que representa a entidade Especialidade	24
Figura 20 - Código de criação tabela Especialidade	24
Figura 21 - Tabela que representa a entidade Consulta	25
Figura 22 - Trigger meterDuracao	25
Figura 23 - Código criação tabela Consulta	25
Figura 24 - Tabela que representa a entidade PUC.....	26
Figura 25 - Trigger precoConsulta.....	26
Figura 26 - Código criação tabela PUC	26
Figura 27 - Tabela que representa a entidade Receita	27
Figura 28 - Código criação tabela Receita	27
Figura 29 - Tabela que representa a entidade Medicamento.....	27
Figura 30 - Código criação tabela Medicamento	28
Figura 31 - Tabela que representa a entidade Medicamento_Receita.....	28
Figura 32 - Código criação tabela Medicamento_Receita.....	28
Figura 33 - Procedure MarcarConsulta	31
Figura 34 - Procedure InserirPS	31
Figura 35 - Procedure RemarcarConsulta	32
Figura 36 - Vista vwMedicamentos.....	34
Figura 37 - Vista vwUtentes.....	35
Figura 38 - Vista listaObito	35

Figura 39 - Código de criação utilizador Médico	36
Figura 40 - Estrutura base para o sistema de dados NoSQL.....	39
Figura 41 - Exemplo de um documento na coleção Utente.....	41
Figura 42 - Exemplo de um documento na coleção Profissionais.....	41
Figura 43 - Exemplo de um documento na coleção Consultas.	42
Figura 44 - Questão 1.....	43
Figura 45 - Questão 2.....	43
Figura 46 - Questão 3.....	43
Figura 47 - Questão 4.....	43
Figura 48 - Questão 5.....	43
Figura 49 - Questão 6	43
Figura 50 - Questão 7.....	43
Figura 51 - Questão 8.....	43
Figura 52 – Conteúdo tabela consulta.....	58
Figura 53 - Conteúdo tabela contacto_PS.....	58
Figura 54 - Conteúdo tabela contacto_Utente	58
Figura 55 - Conteúdo tabela especialidade	59
Figura 56 - Conteúdo tabela incapacidade.....	59
Figura 57 - Conteúdo tabela medicamento_receita.....	59
Figura 58 - Conteúdo tabela medicamento	60
Figura 59 - Conteúdo tabela profissional_saude	60
Figura 60 - Conteúdo tabela receita	60
Figura 61 - Conteúdo tabela puc	61
Figura 62 - Conteúdo tabela utente_incapacitado.....	61
Figura 63 - Conteúdo tabela utente	62

Índice de Tabelas

Tabela 1 - Tabela Utente	12
Tabela 2 - Tabela Incapacidade	12
Tabela 3 - Tabela Utente_Incapacitado	12
Tabela 4 - Tabela Contacto_Utente	13
Tabela 5 - Tabela PUC	13
Tabela 6 - Tabela Profissional_Saude	13
Tabela 7 - Tabela Especialidade	14
Tabela 8 - Tabela Contacto_PS	14
Tabela 9 - Tabela Consulta	14
Tabela 10 - Tabela Medicamento	15
Tabela 11 - Tabela Receita	15
Tabela 12 - Tabela Medicamento_Receita	15
Tabela 13 - Tamanho médio ocupado por cada registo de cada tabela da base de dados	33
Tabela 14 - Quantidade de registos por tabela	33
Tabela 15 - Permissões de um profissional de saúde na BD	36

1. Introdução

1.1. Contextualização

A prevenção da doença é a estratégia mais eficiente a nível de custo-benefício para a gestão da saúde e a doença de uma população. Com a criação de unidades de saúde familiar pretendeu-se chegar a toda a população portuguesa, vacinando, rastreando e tratando fases precoces da doença para prevenir complicações da mesma.

Qualquer cidadão, seja ele nacional ou estrangeiro, tem direito à utilização dos serviços prestados num centro de saúde, serviços esses que variam desde consultas abertas (ou urgentes), consultas de seguimento de patologias específicas, como a hipertensão arterial ou diabetes *mellitus*, acompanhamento da grávida e da criança em crescimento, rastreios segundo o programa nacional de rastreios de doenças neoplásicas, cuidados de enfermagem, vacinação, exames auxiliares de diagnóstico até apoio domiciliário [1].

A gestão de uma unidade hospitalar como um centro de saúde envolve bastante informação, não só acerca dos utentes e seu histórico clínico, mas também relativa à gestão interna dos seus colaboradores e serviços. Por este motivo se infere a importância da implementação de uma base de dados que possa responder às necessidades básicas do dia-a-dia dos colaboradores desta unidade hospitalar.

1.2. Apresentação do Caso de Estudo

Após a proposta da realização de uma Base de Dados, foi decidida a definição de um Centro de Saúde como caso de estudo do projeto.

Um Centro de Saúde é uma unidade básica do SNS para atendimento e prestação de cuidados de saúde à população [1]. De forma a garantir o funcionamento de um Centro de Saúde, são necessários profissionais de saúde (Médicos de Medicina Geral e Familiar, enfermeiros, nutricionistas, etc.), utentes, administrativos, auxiliares. De modo a facilitar a realização desta Base de Dados, foi criada uma versão simplista de um Centro de Saúde, incluindo apenas as entidades imprescindíveis para o caso de estudo – concretamente, algumas das diferentes áreas profissionais presentes numa Unidade de Saúde Familiar (médicos de Medicina Geral e Familiar, internistas, nutricionistas, psicólogos, psiquiatras) e utentes.

1.3. Motivação e Objetivos

O funcionamento sólido de um sistema de gestão de consultas é crucial. A título de exemplo, imaginando que existem três consultas marcadas para as 19h, do mesmo dia, com o mesmo PS. Supondo que cada consulta tem um tempo médio de vinte minutos, significaria que

o último utente a ser atendido teria estado à espera durante um mínimo de quarenta minutos, acrescidos os potenciais atrasos recursivos. A principal motivação na implementação da *BD* será, portanto, prevenir erros de marcação/gestão de tempo e recursos para garantir o funcionamento oleado de todo o Centro de Saúde.

Assim, de modo a ter sucesso na implementação do SGDB relacional, é necessário considerar algumas situações críticas. A realização e posterior utilização da BD possibilitará uma maior facilidade de agendamento e realização de consultas dos diversos utentes.

A elaboração desta base de dados teve, ainda, como objetivo, simplificar o seguimento pormenorizado do estado de saúde de cada pessoa, tendo em conta o seu histórico clínico anteriormente guardado.

Em suma, esta BD deverá permitir realizar diferentes ações no que toca à marcação, cancelamento e atualização de consultas, bem como permitir o registo de novos utentes e profissionais de saúde.

Por fim, a simplicidade de acesso e utilização por parte do utilizador aumentará a viabilidade do projeto.

2. Levantamento e Análise de Requisitos

2.1. Método de levantamento e de análise de requisitos adotado

A fim de desenvolver esta BD da forma mais realista possível, a pesquisa foi iniciada consultando a página do SNS, com o objetivo de perceber como era constituído um centro de saúde, bem como qual a sua orgânica de funcionamento (a título de exemplo e não se esgotando nestes, o horário de funcionamento e qual a finalidade de um centro de saúde). De seguida, recorremos à DGS de forma a conseguir obter dados relativos às metas e objetivos pretendidos (a meta para o número de consultas médicas em 2010 por habitante por ano foi de 4[4]).

Após uma análise relativa ao entendimento de um centro de saúde, o passo seguinte foi a realização de entrevistas estruturadas junto de profissionais de saúde de vários grupos profissionais (médicos e enfermeiros) com o intuito de conhecer o funcionamento interno de um centro de saúde.

Desta forma, foram recolhidas informações sobre o tipo de diagnósticos mais frequentes dos utentes que recorrem ao centro de saúde, tal como a medicação e posologia mais usual. Esta informação, previamente validada, foi transposta para o povoamento da BD. De notar que todos os dados dos utentes e profissionais de saúde são fictícios.

2.2. Requisitos levantados

2.2.1. Requisitos de descrição

- Para que um utente esteja registado na base de dados é necessário ter a informação sobre o seu nome, número de utente, sexo, contacto(s), data de nascimento, morada (rua, cidade e número de porta), profissão e se é ou não dador de sangue;
- Para que um PS seja registado na base de dados, é preciso ter conhecimento do seu nome, sexo, número de cartão de cidadão, contacto(s) e salário base;
- Uma especialidade é constituída por uma descrição e um Id;
- Uma incapacidade é caracterizada por uma descrição e um Id;
- Uma consulta contém um Id, uma hora de entrada/saída, um estado, uma descrição e uma data de marcação;
- Uma receita dispõe de uma data, uma descrição e um Id;
- Um medicamento é caracterizado por um nome, Id e descrição.

2.2.2. Requisitos de exploração

- A base de dados deve permitir marcar consultas;
- Deve permitir desmarcar ou remarcar consultas;
- Deve permitir o registo de novos utentes e profissionais de saúde;
- Deve permitir emitir receitas;
- Saber quais as receitas emitidas numa consulta e para determinado utente;
- Obter uma lista das consultas que o PS vai realizar em determinado dia;
- Conhecer o número total de consultas do PS;
- Ter acesso a uma lista de todos os utentes já falecidos;
- Obter o valor do bónus anual recebido pelo PS, em acréscimo ao seu salário base;

2.2.3 Requisitos de controlo

- A base de dados deve permitir ter conhecimento do falecimento de um utente, desmarcando as consultas que este tinha marcadas;
- A base de dados não deve permitir que sejam agendadas consultas com sobreposição no mesmo profissional de saúde ou para um só utente;
- Não devem ser marcadas consultas antes das oito (8) horas e depois das vinte (20) horas, uma vez que o Centro de Saúde se encontra encerrado;
- Não devem ser marcadas consultas para os dias de Natal e de Ano Novo, dado o encerramento do Centro de Saúde;

2.3. Análise geral dos requisitos

O registo dos profissionais de saúde deve conter os respetivos nome, sexo, contacto(s), salário base fixo e o número de cartão de cidadão, este último que irá servir como identificação capaz de distinguir os diferentes PS, tendo em conta que este número é pessoal e único. Além disso, cada PS deve ter especificado qual a sua profissão e especialidade, à qual corresponderá um ID e uma descrição.

Aquando do registo dos utentes, torna-se inevitável a criação de um Id, que irá identificar cada utente. Para a concretização do seu registo, é requisitado o nome do utente, sexo, data de nascimento, profissão, contacto(s), morada (contendo informações detalhadas sobre a cidade, rua e número de porta), bem como a indicação de ser dador de sangue ou não. Cada utente deve, ainda, informar se possui qualquer incapacidade, à qual será feita uma breve descrição e atribuído um Id. Uma vez que alguns utentes podem apresentar mais do que uma incapacidade, é essencial estabelecer uma ligação entre o utente e as suas diversas incapacidades.

Como meio de verificação do atingimento das metas, preconizadas no Plano Nacional de Saúde [2], por parte dos PS, é crucial poder visualizar o número de consultas realizadas no intervalo de tempo estipulado. Desta forma, é necessário ter registada a data da realização da consulta, bem como a hora de entrada e de saída do utente (permitindo ter controlo sobre a duração média de cada consulta realizada), a descrição da consulta, devendo conter observações tomadas pelo PS, e um Id que diferencia cada consulta entre si. Deve, ainda, estar presente o estado da consulta, uma vez que esta pode ter sido marcada, iniciada, realizada ou cancelada. Estas metas vão permitir acrescentar um valor ao salário de cada PS, se atingidas.

As receitas emitidas durante as consultas contêm também um Id, uma descrição, o Id da consulta em que foi emitida e também a data da sua emissão. Sabendo que na mesma consulta podem ser emitidas várias receitas, é estabelecida uma ligação entre a consulta e as eventuais receitas. Cada receita é, por sua vez, constituída por um ou vários, medicamentos, os quais possuem um Id, um nome e também uma descrição. A ligação estabelecida entre os medicamentos e a respetiva receita possui a posologia referente a cada medicamento nela contido.

Por fim, de forma a obter o preço que cada utente terá que pagar, irá ser tida em conta a situação de dador, ou não, de sangue, e a faixa etária do utente. Caso o utente possua uma idade superior a sessenta e cinco (65) anos, não terá que pagar a consulta. No caso de ser dador, o preço final será de quatro (4) euros. Em qualquer outro caso, estipula-se um preço fixo de cinco (5) euros.

A BD será realizada de acordo com as necessidades do Centro de Saúde e, por isso, terá em conta o seu horário de funcionamento (com início às 08:00 horas e término às 20:00 horas, encerrado nos dias de Natal e Ano Novo). Desta modo, irá permitir que cada utente tenha as suas consultas marcadas, dando-lhes a possibilidade de desmarcar e também remarcar as mesmas, pelo que não irá autorizar o agendamento de consultas com sobreposição para o mesmo PS, nem marcação de consultas com pouco distanciamento para um mesmo utente. Desta forma, permitirá, também, atualizar o estado das consultas, já que será necessário contabilizar o número de consultas realizadas e agendadas por determinado PS. Além disso, a BD deverá gerir o falecimento de um utente, pelo que este será marcado como falecido, sendo que as consultas que este tinha previamente marcadas serão canceladas.

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

O primeiro passo para a criação de uma BD é a elaboração do modelo conceptual. Utilizando um Diagrama Entidades-Relacionamentos (Diagrama ER), composto, tal como o nome diz, pelas entidades, atributos e pelos relacionamentos que existem entre si.

As entidades são representadas pelos retângulos, sendo que estas possuem atributos, representados por elipses. Cada entidade tem uma chave-primária, ou seja, um atributo único e distinto de todos os outros possuídos por outras entidades. Esta chave é representada com uma coloração mais escura e possui texto sublinhado. Por exemplo, a chave-primária de um *Utente* é o seu *Número de Utente*. Além disso, os relacionamentos são representados por losangos, que podem ser do tipo “1:1”, “1:N” ou “N:N”.

3.2. Identificação e caracterização das entidades

Existem sete entidades cruciais para a resolução do problema, sendo estas: utente, profissional de saúde, consulta, incapacidade, especialidade, receita e medicamento.

Um utente é quem se desloca até ao centro de saúde, de forma a ser analisado por uma entidade especializada na área da saúde. É caracterizado pelo seu número de utente, nome, data de nascimento, sexo, contactos, entre outros. Tendo em conta que a única característica que este possui que é, realmente, única, é o seu número de utente, esta é a sua chave primária. Um dos seus contactos poderia ser considerado uma chave candidata, mas pode existir partilha do mesmo, pelo que foi desconsiderada a hipótese.

No caso da incapacidade, restrição física e/ou psicológica, a sua designação poderia efetuar o papel de chave, no entanto, torna-se mais simples comparar e procurar inteiros, daí o uso de um identificador como chave primária.

O profissional de saúde tem associado um ou vários contactos de modo a ser possível contactá-lo em caso de necessidade; um nome, um sexo e um salário base mensal, para efeitos de finanças. Note-se que o mesmo apenas é especialista numa área. Como este possui um número único no seu cartão de cidadão e, tendo em conta as restrições dos contactos mencionadas anteriormente, esta é a chave primária escolhida.

A consulta caracteriza o momento da observação do utente por parte da entidade de saúde. Sabendo que os seus atributos principais são a sua data de marcação e identificador, sendo este último a chave primária, já que poderá haver mais que uma consulta a ocorrer em simultâneo.

A especialidade representa a área de especialização do PS. Como no caso da incapacidade, esta apenas possui uma descrição, que neste caso representa o nome, e um identificador. Por motivos de eficiência, optou-se colocar o Id como chave primária.

Sendo a receita um documento onde estão listados os medicamentos a adquirir, terá que possuir, além desta lista, uma data de emissão e uma descrição, composta por anotações tiradas pelo PS. Com o objetivo de aumentar a eficiência de procura na BD em mente, foi criado, mais uma vez, um Id como chave primária.

Para concluir, um medicamento representa um fármaco a ser tomado pelo utente, necessitando de ter uma posologia e um nome. Sendo que o nome poderia representar a sua chave, baseando a escolha nos argumentos até agora mencionados, a chave primária escolhida foi o Id.

3.3. Identificação e caracterização dos relacionamentos

Sabendo que o recurso de um utente a uma consulta exige a presença de um PS, tem de existir um relacionamento entre estas três entidades. Um utente pode marcar mais que uma consulta e visitar vários profissionais de saúde. Um PS observa também vários utentes. Assim, tem de existir um relacionamento triplo “N:N:N” entre estas entidades.

O relacionamento entre o utente e a incapacidade será do tipo “N:N”: Um utente pode ser portador de uma ou mais incapacidades. Por outro lado, uma incapacidade pode ser comum a várias pessoas.

Um PS só pode ser especializado numa área. Por exemplo, apenas pode ser “médico de Medicina Geral e Familiar”, “nutricionista”, “médico internista”, “psicólogo”, “psiquiatra”, etc.. Deste modo, tem de existir um relacionamento “N:1” entre os PS e a sua especialidade.

Outro relacionamento existente é entre uma consulta e as receitas emitidas na mesma. Tendo em conta que numa consulta não há limite do número de receitas emitidas e uma receita está associada a apenas uma consulta, a cardinalidade deste relacionamento é de “1:N”.

Por fim, cada receita pode conter N medicamentos, tal como um medicamento pode ser mencionado em várias receitas, sendo, portanto, um relacionamento “N:N”.

3.4. Identificação e caracterização dos Atributos com as Entidades e Relacionamentos

Começando pelos atributos relativos a entidades, tem-se: um utente necessita de deter um número de utente, ou seja, o número pelo qual é caracterizado na instituição, uma morada, definida por cidade, rua e número, uma faixa etária, relativa à sua data de nascimento, uma profissão, um sexo, podendo ser ‘M’ de masculino ou ‘F’ de feminino, um nome, a indicação sobre se é ou não dador de sangue, uma data de óbito (que não possui valor até a morte do utente ocorrer), e uma lista dos seus contactos. Cada incapacidade apresenta um identificador e uma descrição, que, neste caso, representa o seu nome. Um PS está definido pelo seu cartão de cidadão, nome, sexo, salário base e uma lista de contactos. Tal como no caso da incapacidade, a especialidade apenas possui um identificador e uma descrição. O grupo de

entidades final está interligado à consulta, sendo que esta apresenta um identificador, uma data, na forma de dia e hora relativa à marcação da consulta, uma hora de entrada e de saída que dão fruto a uma duração e que podem ser utilizadas para conhecer o atraso relativo à marcação, um estado, podendo ser “cancelada”, “marcada”, “iniciada” e “realizada” e uma descrição com informações adicionais. Uma receita tem de conter um identificador, uma data, relativa à sua emissão, e uma descrição. Por fim, um medicamento possui um nome, uma descrição adicional e um identificador.

Já os atributos relativos a relacionamentos são visíveis na relação entre o utente, PS e consulta, onde um preço é pago pelo primeiro como forma de remuneração pelo serviço prestado. O outro atributo presente num relacionamento é relativo à posologia de cada medicamento na receita, ou seja, a sua dosagem diária.

3.5. Apresentação e explicação do diagrama ER

Como é visível na Figura 1, o diagrama tem em consideração todas as características até então mencionadas, permitindo ao utente ter diversos atributos, entre os quais o seu nome, profissão, sexo, morada, número de utente, etc., além de ser possível que este tenha mais que uma incapacidade.

Além do mais, permite que numa consulta sejam emitidas todas as receitas necessárias, sendo que estas permitem a prescrição de diversos medicamentos. A BD concede uma única especialidade ao profissional de saúde, bem como os atributos por este requisitados.

Finalmente, uma consulta irá relacionar o PS com o utente observado, tendo a si associada as receitas em questão, um identificador, uma data e hora de marcação, a duração (derivada da diferença entre a hora de entrada e a hora de saída), uma descrição e o seu estado, que permitirá saber quais consultas estão a ser efetuadas num determinado momento, quais foram canceladas, etc..

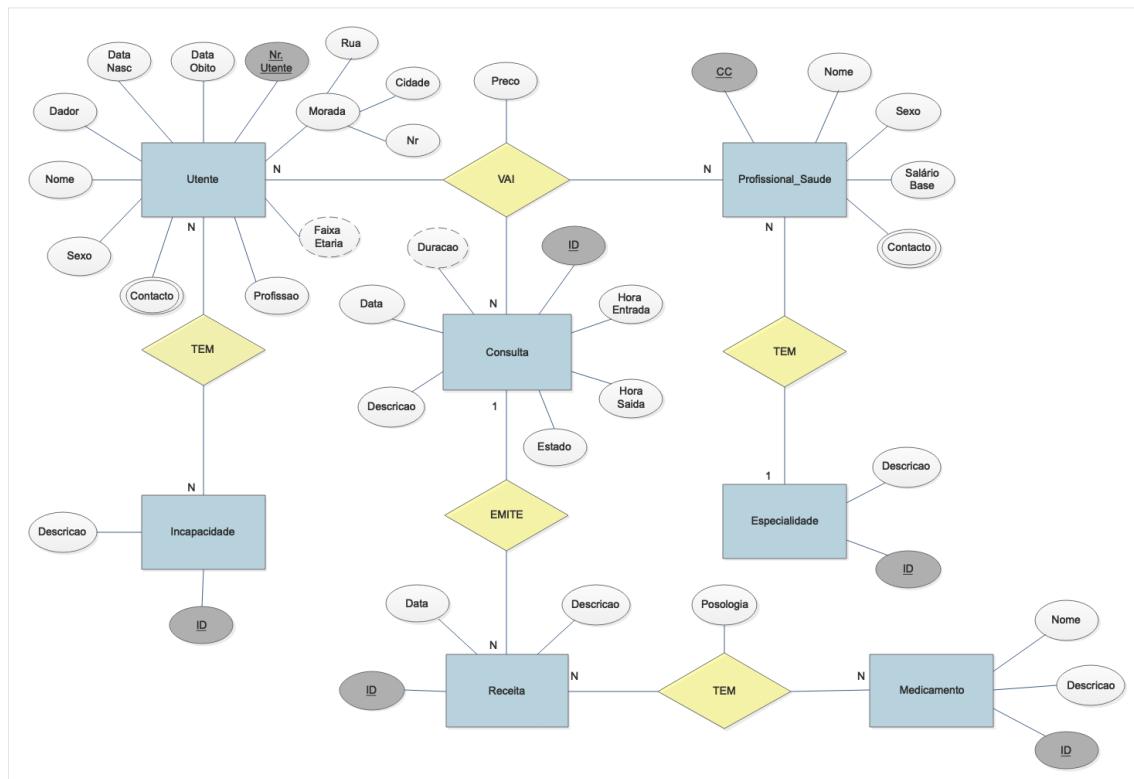


Figura 1 - Modelo Conceptual da BD

3.6. Validação do modelo de dados com o utilizador

Finita a modelação do modelo conceitual, é crucial verificar a possibilidade de realizar transações e interrogações, provando o funcionamento correto do modelo. Caso existam impossibilidades em realizar operações, será necessário reformular o modelo de modo a permitir um desempenho adequado. Existem duas (2) formas de assegurar a validade de um modelo conceitual [3], sendo estas “Descrição da Transação” e “Uso de Caminhos de Transação”. Optou-se pela utilização da primeira forma. Assim, é necessário verificar que toda a informação fornecida pelas entidades, relacionamentos e atributos é fornecida pelo modelo.

Destacam-se alguns exemplos de transações e interrogações realizadas no projeto:

- Obter a listagem das receitas emitidas numa determinada consulta: Como existe um relacionamento “1:N” entre as entidades *Consulta* e *Receita*, é possível efetuar esta interrogação;
- Obter a listagem das consultas que um PS irá realizar num determinado dia: Tendo em conta a existência de um relacionamento ternário entre *Consulta*, *Utente* e *Profissional de Saúde* e, sabendo que um atributo da consulta é a data de marcação, esta interrogação é, também, possível;
- Determinar o valor do bónus anual de um determinado PS: Assuma-se que por cada utente que visite o mesmo médico, quatro (4) ou mais vezes num ano, o último recebe um bónus de cinquenta (50) euros nesse ano. Assim, utilizando o relacionamento ternário mencionado acima, é possível calcular a quantidade de vezes que o utente X visitou o profissional Y no ano Z. Não esquecer que é crucial o uso do atributo *Data* e *Estado*, pois apenas se têm em consideração consultas efetuadas.
- Inserir um Utente, sendo que é necessária a informação sobre o seu nome, número de utente, sexo, contacto(s), data de nascimento, morada (rua, cidade e número de porta), profissão e se é ou não dador: Para efetuar esta transação é necessário criar um novo *Utente*, sendo que este será alojado na respetiva tabela. Como o utente possui todos os atributos acima mencionados, esta transação é possível.
- Marcar uma consulta, sendo que esta deverá possuir um identificador, uma data de marcação, hora de entrada, hora de saída, um estado, uma duração e uma descrição: Tendo em conta os atributos associados a esta entidade, bem como o facto de esta estar interligada com um *Utente*, um PS e as receitas emitidas pela mesma, esta transação é, de igual modo, possível.

Encontrando-se a validação efetuada relativamente a todos os requisitos indicados pelo utilizador, poder-se-á, de seguida, realizar o modelo lógico.

4. Modelação Lógica

A segunda fase do desenvolvimento de uma BD envolve a criação de um Modelo Lógico, tendo como base o Modelo Conceptual desenvolvido anteriormente. Este é obtido por processos de normalização, que permitem a validação do modelo consoante os requisitos dos utilizadores, obtendo-se, assim, um conjunto de tabelas que permitem um correto registo dos dados. O ML deste projeto encontra-se na Figura 2 e foi representado utilizando o MySQL Workbench.

Em cada tabela existe um atributo que atua como chave primária, podendo ainda permitir a existência de chaves estrangeiras, sendo estas chaves primárias noutras tabelas. Nas tabelas identificam-se ainda os atributos, assim como o seu tipo de dados e tamanho.

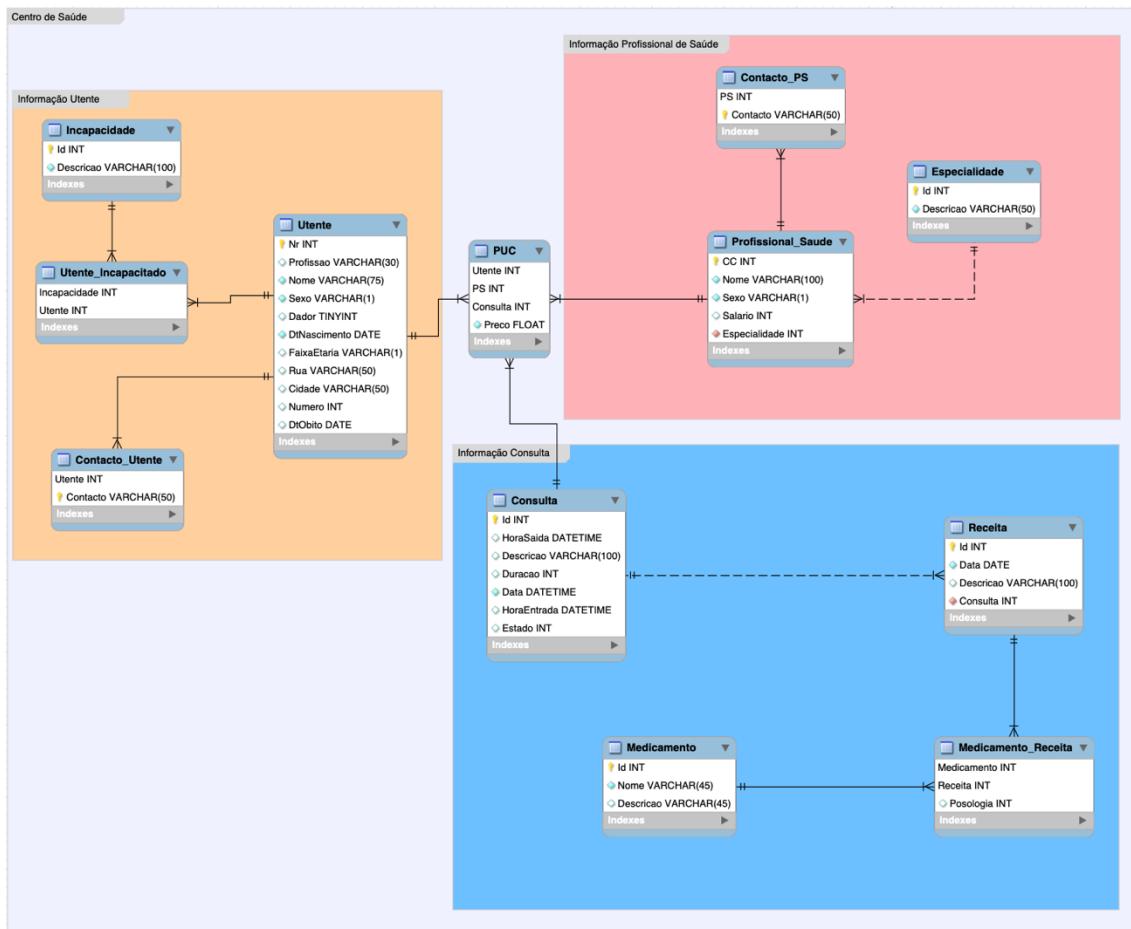


Figura 2 - Modelo Lógico da BD

4.1. Construção do modelo de dados lógico

4.1.1. Tabela Utente

Nesta tabela estão identificados todos os utentes que se encontram registados na instituição em questão. É composta por onze (11) atributos, sendo que o número do próprio (*Nr*) é único a cada pessoa, representando, portanto, a chave primária desta tabela. O atributo *DtObito* é referente à data do falecimento do utente, tomando o valor de *NULL* enquanto este estiver vivo. Os restantes atributos dizem respeito aos dados pessoais de cada utente.

Atributos	Chave Primária	Chave Estrangeira
Nr	X	
Profissao		
Nome		
Sexo		
Dador		
DNascimento		
FaixaEtaria		
Rua		
Cidade		
Numero		
DtObito		

Tabela 1 - Tabela Utente

4.1.2. Tabela Incapacidade

A tabela em causa apresenta as incapacidades relativas a cada utente. O atributo *Id* representa o identificador da mesma, sendo assim a sua chave primária. A *Designação* possui o nome da respetiva incapacidade.

Atributos	Chave Primária	Chave Estrangeira
Id	X	
Designação		

Tabela 2 - Tabela Incapacidade

4.1.3. Tabela Utente_Incapacitado

Esta tabela é composta por dois (2) atributos. Juntos fazem uma chave primária composta. Esta tabela estabelece a ligação entre um utente e a(s) sua(s) incapacidade(s).

Atributos	Chave Primária	Chave Estrangeira
Incapacidade	X	X
Utente	X	X

Tabela 3 - Tabela Utente_Incapacitado

4.1.4. Tabela Contacto_Utente

Aqui é feito o relacionamento entre o utente e o seu(s) contacto(s), sendo que o Nr do *Utente* e o contacto representam a chave primária composta. De notar que o Nr do utente é uma chave estrangeira.

Atributos	Chave Primária	Chave Estrangeira
Utente	X	X
Contacto	X	

Tabela 4 - Tabela Contacto_Utente

4.1.5. Tabela PUC

Nesta tabela encontra-se a junção relacional das tabelas *Utente*, *Consulta* e *Profissional_Saude*, sendo que as chaves primárias destas três (3) fazem a chave primária composta da tabela. Além disso, existe um atributo *Preço*, representativo do valor a pagar pelo serviço.

Atributos	Chave Primária	Chave Estrangeira
Utente	X	X
PS	X	X
Consulta	X	X
Preco		

Tabela 5 - Tabela PUC

4.1.6. Tabela Profissional_Saude

A tabela Profissional_Saude apresenta todos os profissionais de saúde, possuindo cinco (5) atributos. Semelhante ao caso do *Utente*, o atributo CC é um número de identificação único para cada PS, sendo, portanto, a sua chave primária. Esta tabela apresenta uma chave estrangeira relativa à especialidade do profissional. Os restantes atributos são relativos a dados pessoais.

Atributos	Chave Primária	Chave Estrangeira
CC	X	
Nome		
Sexo		
Salario		
Especialidade		X

Tabela 6 - Tabela Profissional_Saude

4.1.7. Tabela Especialidade

Neste caso, a tabela possui um atributo identificador (*Id*), sendo a sua chave primária, e um atributo descritor (*Descrição*), que conterá o nome da especialidade em questão.

Atributos	Chave Primária	Chave Estrangeira
Id	X	
Descricao		

Tabela 7 - Tabela Especialidade

4.1.8. Tabela Contacto_PS

Tal como no caso da tabela *Contacto_Utente*, é feito o relacionamento entre o profissional de saúde e o seu(s) contacto(s), sendo que o CC do PS e o contacto representam a chave primária composta. Além disso, o CC do PS é uma chave estrangeira.

Atributos	Chave Primária	Chave Estrangeira
Contacto	X	
PS	X	X

Tabela 8 - Tabela Contacto_PS

4.1.9. Tabela Consulta

A tabela *Consulta* possui uma chave primária na forma de identificador (*Id*), um atributo relativo à data de marcação (*Data*), uma hora de entrada (*HoraEntrada*) e saída (*HoraSaida*), com consequente duração. Possui, ainda, uma descrição relativa a observações anotadas no decorrer da consulta e um estado na forma de inteiro, podendo tomar quatro (4) valores: 1 – *Marcada*; 2 – *Iniciada*; 3 – *Realizada*; 4 – *Cancelada*.

Atributos	Chave Primária	Chave Estrangeira
Id	X	
HoraSaida		
Descricao		
Duracao		
HoraEntrada		
Data		
Estado		

Tabela 9 - Tabela Consulta

4.1.10. Tabela Medicamento

Nesta tabela, um Medicamento é caracterizado pelo seu identificador (*Id*), sendo este a sua chave primária. Contém, também, um nome e uma descrição, relativa à formulação e dose do fármaco (por exemplo “10mg comprimido” ou “supositório”).

Atributos	Chave Primária	Chave Estrangeira
Id	X	
Nome		
Descricao		

Tabela 10 - Tabela Medicamento

4.1.11. Tabela Receita

Esta tabela possui um identificador (*Id*), uma data, relativa à emissão do documento médico, uma descrição que contém o método de toma da medicação e o identificador da consulta em que foi emitida.

Atributos	Chave Primária	Chave Estrangeira
Id	X	
Data		
Descricao		
Consulta		X

Tabela 11 - Tabela Receita

4.1.12. Tabela Medicamento_Receita

Este relacionamento liga uma receita aos medicamentos existentes na mesma. Assim, o *Id* do *Medicamento* e o *Id* da *Receita*, juntos, constituem a chave primária composta da tabela. Fora isso, apresenta a posologia, ou seja, o número de tomas diárias do medicamento, indicado pelo profissional de saúde.

Atributos	Chave Primária	Chave Estrangeira
Medicamento	X	X
Receita	X	X
Posologia		

Tabela 12 - Tabela Medicamento_Receita

4.2. Validação do Modelo através da Normalização

Após a implementação do modelo, foi necessário validar o mesmo em relação à normalização, de maneira a garantir a inexistência de anomalias entre as relações.

Para garantir a primeira forma normal é necessário que a interseção entre cada linha e as colunas contenha um único valor, de modo a assegurar a atomicidade dos valores, aspecto fulcral para a garantia da não existência de dados repetidos na BD. Quer isto dizer que, no caso de estudo, os atributos compostos devem ser separados, como no caso da morada, os multivvalorados devem ser colocados numa nova tabela com uma chave primária composta, sejam exemplos as tabelas de contactos.

Após garantida esta característica, é necessário asseverar a segunda forma normal: qualquer atributo que não uma chave primária tem que ser totalmente dependente da chave primária da tabela a que este pertence. Quer isto dizer que, caso exista algum atributo numa tabela que não dependa diretamente da chave primária dessa, é necessária a alocação desse atributo numa nova tabela onde exista uma dependência direta com a chave primária da nova tabela. Cumprido este requisito é garantida a não redundância de dados no modelo.

Por fim é necessário assegurar que o modelo cumpre os requisitos da terceira forma normal, sendo para isso necessária a garantia de que nenhum atributo descritor, não chave, dependa funcionalmente de outro atributo não chave, isto é, não existam dependências transitivas.

Após análise do modelo até então efetuado, é possível concluir que este cumpre todas estas formas normais, garantindo assim uma não redundância dos dados presentes no modelo.

4.3. Validação do Modelo com as interrogações do Utilizador

Depois de garantida a normalização do modelo é necessário confirmar que o mesmo cumpre todos os requisitos expressos inicialmente, correspondendo, assim, ao que era pretendido pelos utilizadores.

Ora, a obtenção da listagem das receitas emitidas numa determinada consulta é possível, já que a tabela *Consulta* está ligada à *Receita*, fruto da existência de um relacionamento “1:N” entre as mesmas.

Já a obtenção da lista de consultas que um PS irá realizar num determinado dia, bem como a listagem das consultas associadas a um dado PS são conseguidas pela ligação entre a tabela *Profissional_Saude* com *PUC*, que está ligada à *Consulta*. A tabela *PUC* permite interligar um PS, uma *Consulta* e, ainda, um *Utente*, graças à sua chave primária composta, feita pela junção das chaves primárias das três (3) tabelas mencionadas.

A determinação do bónus anual de um PS é possível com o auxílio da tabela *PUC*, relacionando as três entidades chave para o problema em questão. Note-se que é necessário conhecer a data de marcação da consulta, bem como o seu estado, com a finalidade de saber se ela será utilizada para a contagem. Além disso, é necessário agrupar as consultas por cada

utente, de modo a apurar quantos visitaram o PS em questão quatro (4) ou mais vezes durante o ano.

O preço de uma *Consulta* deve variar consoante a faixa etária e o facto de *Utente* ser ou não dador de sangue, sendo isto garantido através do relacionamento entre as tabelas *PUC*, *Utente* e *Consulta*.

É, ainda, requerido que não exista sobreposição de consultas, ou seja, um utente e um *PS* não poderão ter a si associadas mais que uma consulta num intervalo de quinze (15) minutos. Este requisito garante-se com o auxílio do atributo *Data* na tabela *Consulta*, associada, tanto ao *Utente*, como ao *PS*, pela tabela *PUC*.

Outra interrogação pedida pelo cliente é que não existam géneros inválidos, ou seja, apenas exista 'F' para feminino ou 'M' para masculino. Mais uma vez é garantido, quer na tabela *Utente*, quer na tabela *PS*, graças ao atributo *Sexo*.

Por fim, o acesso a uma lista de todos os utentes já falecidos é garantido com a tabela *Utente*, com auxílio do atributo *DtObito* que indica a data da morte do utente. Caso o utente se encontre vivo, esta fica com o valor *NULL*.

4.4. Validação do modelo com as transações estabelecidas

Garantida a validação do modelo com as interrogações do utilizador, é necessário validar o mesmo em relação às transações estabelecidas.

Uma transação é uma sequência de operações executadas como se fossem uma só, que alteram o conteúdo da BD.

A marcação, desmarcação e remarcação de consultas é assegurada na tabela *Consulta*, graças ao atributo *Estado*, pois esta indica o estado de uma consulta, que pode variar entre *marcada*, *iniciada*, *realizada* e *cancelada*. A adição só é efetuada se ainda não existir uma linha da tabela com a mesma chave primária.

O registo de novos utentes e profissionais de saúde é assegurado pelas tabelas *Utente* e *Profissional_Saude*, respetivamente, sendo garantida a não existência de entidades repetidas.

A emissão de receitas é garantida na tabela *Receita*, sendo automaticamente associadas à consulta onde foram emitidas. Graças a isto, é possível conhecer todas as receitas emitidas numa consulta. Além disso, como um *Utente* está interligado às consultas que vai, pela tabela *PUC*, as receitas do mesmo estarão, também, a si relacionadas.

Por fim, a ocorrência do óbito de um utente implica que sejam desmarcadas todas as consultas que o mesmo tivesse marcadas, algo possível devido à ligação entre *Utente* com *Consulta* pela tabela *PUC*. Ademais, a data do óbito fica associada ao *Utente* no atributo *DtObito*.

Efetuada a validação de todas as transações necessárias e requeridas, resta rever o modelo lógico com o utilizador, na tentativa de confirmar se completa as necessidades do mesmo.

4.5. Revisão do modelo lógico com o utilizador

Efetuada a validação de todas as transações necessárias e requeridas, é necessário rever o modelo lógico com o utilizador, na tentativa de confirmar se completa as necessidades do mesmo, ou seja, se o modelo permite o funcionamento normal da unidade hospitalar.

Para tal foi necessário reunir com os futuros utilizadores da BD, para que estes avaliassem o modelo e confirmassem que este modela com exatidão a situação pretendida. Através de uma reunião com os funcionários da unidade hospitalar, foi confirmado que o modelo representava o pretendido para a unidade.

Garantida a aprovação, foi iniciado desenvolvimento da última fase do projeto, o modelo físico, tendo em conta os respetivos modelos conceitual e lógico como base.

5. Implementação Física

5.1. Seleção do sistema de gestão de bases de dados

Uma base de dados é gerida por um Sistema de Gestão de Bases de Dados, que condiciona a forma como se cria, acede, modifica e elimina informação. Desta forma, um SGBD deve permitir definir a estrutura dos dados mais adaptada ao problema, bem como definir permissões de acesso aos dados; permitir a consulta eficiente dos dados, independentemente de como estes estiverem armazenados; possibilitar que a base de dados possa ser recuperada; proporcionar operações simultâneas sobre os dados por parte de vários utilizadores [5].

Para a concretização desta BD, foi selecionado o SGBD relacional MySQL, que utiliza a linguagem SQL, por ser rápido, seguro e simples de ser utilizado, desenvolvido para conseguir lidar de forma eficaz com bases de dados de grande dimensão [6].

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

Aquando da implementação do modelo lógico, foram criadas tabelas, cada uma respeitante a determinada entidade, cujas colunas representam os atributos que cada apresenta. O modelo físico, implementado através do modelo lógico, é suportado por estas tabelas, compostas por chaves primárias, chaves estrangeiras e atributos, que definem cada entidade.

Tables
Consulta
Contacto_PS
Contacto_Utente
Especialidade
Incapacidade
Medicamento
Medicamento_Receita
Profissional_Saude
PUC
Receita
Receita_Consulta
Utente
Utente_Incapacitado

Figura 4 - Tabelas do Modelo Físico

Tables
Consulta
Columns
Id
HoraSaida
Descricao
Duracao
Data
HoraEntrada
Estado
Indexes
Foreign Keys
Triggers
Contacto_PS

Figura 3 - Conteúdo da tabela Consulta no Modelo Físico

5.2.1. Tabela Utente

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Nr	INT	◊	✓	✓	□	□	□	□	□	□
Profissao	VARCHAR(30)	◊	□	□	□	□	□	□	□	□
Nome	VARCHAR(75)	◊	□	✓	□	□	□	□	□	□
Sexo	VARCHAR(1)	◊	□	✓	□	□	□	□	□	□
Dador	TINYINT	◊	□	□	□	□	□	□	□	□
DtNascimento	DATE	◊	□	✓	□	□	□	□	□	□
FaixaEtaria	VARCHAR(1)	◊	□	□	□	□	□	□	□	□
Rua	VARCHAR(50)	◊	□	□	□	□	□	□	□	□
Cidade	VARCHAR(50)	◊	□	□	□	□	□	□	□	□
Numero	INT	◊	□	□	□	□	□	□	□	□
DtObito	DATE	◊	□	□	□	□	□	□	□	□

Figura 5 - Tabela que representa a entidade Utente

Um utente é identificado pelo Nr, sendo este atributo a chave primária, com o objetivo de garantir que não existem dois números de igual valor para diferentes utentes, esta chave é não nula e única.

Em relação ao registo do utente, é indispensável ter as informações sobre o seu nome, sexo e data de nascimento, pelo que estes atributos são definidos como não nulos.

De forma a calcular a faixa etária do utente (atributo derivado), foi definido o *Trigger AtualizarFaixaEtaria*, que é executado antes da inserção do utente na tabela.

```
DELIMITER $$  
CREATE TRIGGER AtualizaFaixaEtaria BEFORE INSERT ON Utente  
FOR EACH ROW  
BEGIN  
    IF (TIMESTAMPDIFF(YEAR, NEW.DtNascimento, CURDATE()) < 18) THEN SET NEW.FaixaEtaria = 'J';  
    ELSE IF (TIMESTAMPDIFF(YEAR, NEW.DtNascimento, CURDATE()) > 65) THEN SET NEW.FaixaEtaria = 'S';  
    ELSE SET NEW.FaixaEtaria = 'A';  
    END IF;  
    END IF;  
END $$;  
DELIMITER ;
```

Figura 6 - Trigger AtualizarFaixaEtaria

De modo a garantir que não é introduzido um valor de sexo errado, foram definidos os Triggers *GenderSafe* e *GenderSafePS*, que não permitem que seja introduzido um caracter diferente de 'F' ou 'M', tanto para um utente, como para um PS, respetivamente.

```
-- Implementação de um trigger que garante que não é inserido nenhum sexo aquando o registo do Utente que não seja M ou F
DELIMITER $$ 
CREATE TRIGGER GenderSafe BEFORE INSERT ON Utente
FOR EACH ROW
BEGIN
    IF (new.Sexo <> 'F' AND new.Sexo <> 'M')
    THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Sexo escolhido não é válido, use F ou M';
    END IF;
END $$ 
DELIMITER $$

-- Implementação de um trigger que garante que não é inserido nenhum sexo aquando o registo do PS que não seja M ou F
DELIMITER $$ 
CREATE TRIGGER GenderSafePS BEFORE INSERT ON Profissional_Saude
FOR EACH ROW
BEGIN
    IF (new.Sexo <> 'F' AND new.Sexo <> 'M')
    THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Sexo escolhido não é válido, use F ou M';
    END IF;
END $$ 
DELIMITER $$
```

Figura 7 - Triggers GenderSafe e GenderSafePS

O código SQL para a criação da tabela *Utente* é o seguinte:

```
CREATE TABLE IF NOT EXISTS `centrosaude`.`Utente` (
  `Nr` INT NOT NULL,
  `Profissao` VARCHAR(30) NULL,
  `Nome` VARCHAR(75) NOT NULL,
  `Sexo` VARCHAR(1) NOT NULL,
  `Dador` TINYINT NULL,
  `DtNascimento` DATE NOT NULL,
  `FaixaEtaria` VARCHAR(1) NULL,
  `Rua` VARCHAR(50) NULL,
  `Cidade` VARCHAR(50) NULL,
  `Numero` INT NULL,
  `DtObito` DATE NULL,
  PRIMARY KEY (`Nr`))
ENGINE = InnoDB;
```

Figura 8 - Código de criação tabela Utente

5.2.2. Tabela Contacto_Utente

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Utente	INT	◊	✓	✓	□	□	□	□	□	
Contacto	VARCHAR(50)	◊	✓	✓	□	□	□	□	□	

Figura 9 - Tabela que representa a entidade Contacto_Utente

Dado que os contactos de um utente são um atributo multivalue, foi criada uma tabela *Contacto_Utente* que contém uma chave primária composta pelo *Utente* (INT) e *Contacto* (VARCHAR). Como se trata de uma chave primária, os seus valores são não nulos e únicos, pois são distintos entre si.

```

CREATE TABLE IF NOT EXISTS `centrosaude`.`Contacto_Utente` (
  `Utente` INT NOT NULL,
  `Contacto` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`Utente`, `Contacto`),
  CONSTRAINT `fk_Contacto_Utente_Utente`
    FOREIGN KEY (`Utente`)
    REFERENCES `centrosaude`.`Utente` (`Nr`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 10 - Código de criação tabela Contacto_Utente

5.2.3. Tabela Incapacidade

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Id	INT	◊	✓	✓	□	□	□	□	□	
Descricao	VARCHAR(100)	◊	□	✓	□	□	□	□	□	

Figura 11 - Tabela que representa a entidade Incapacidade

Uma *Incapacidade* é identificada por um Id, constituindo a chave primária da tabela, sendo não nulo e único. O atributo descrição é, também, não nulo, já que é necessária para saber qual o nome da incapacidade.

```

CREATE TABLE IF NOT EXISTS `centrosaude`.`Incapacidade` (
  `Id` INT NOT NULL,
  `Descricao` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;

```

Figura 12 - Código de criação tabela Incapacidade

5.2.4. Tabela Utente_Incapacitado

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Incapacidade	INT	◊	✓	✓	□	□	□	□	□	
Utente	INT	◊	✓	✓	□	□	□	□	□	

Figura 13 - Tabela que representa a entidade Utente_Incapacitado

Como o relacionamento entre *Utente* e *Incapacidade* é “N:N”, foi criada uma tabela *Utente_Incapacitado*, cuja chave primária composta contém as chaves da *Incapacidade* e do *Utente*.

```

CREATE TABLE IF NOT EXISTS `centrosaude`.`Utente_Incapacitado` (
  `Incapacidade` INT NOT NULL,
  `Utente` INT NOT NULL,
  PRIMARY KEY (`Incapacidade`, `Utente`),
  INDEX `fk_Incapacidade_has_Utente_Utente1_idx` (`Utente` ASC) VISIBLE,
  INDEX `fk_Incapacidade_has_Utente_Incapacidade1_idx` (`Incapacidade` ASC) VISIBLE,
  CONSTRAINT `fk_Incapacidade_has_Utente_Incapacidade1`
    FOREIGN KEY (`Incapacidade`)
      REFERENCES `centrosaude`.`Incapacidade` (`Id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  CONSTRAINT `fk_Incapacidade_has_Utente_Utente1`
    FOREIGN KEY (`Utente`)
      REFERENCES `centrosaude`.`Utente` (`Nr`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 14 - Código de criação tabela Utente_Incapacitado

5.2.5. Tabela Profissional_Saude

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
CC	INT	◊	✓	✓	□	□	□	□	□	
Nome	VARCHAR(100)	◊	□	✓	□	□	□	□	□	
Sexo	VARCHAR(1)	◊	□	✓	□	□	□	□	□	
Salario	INT	◊	□	□	□	□	□	□	□	
Especialidade	INT	◊	□	✓	□	□	□	□	□	

Figura 15 - Tabela que representa a entidade Profissional_Saude

Um profissional de saúde é identificado pela sua chave primária CC, que é um inteiro não nulo e único, distinguindo-o de todos os outros profissionais de saúde. Quando se regista um PS, é necessário ter acesso aos dados sobre o seu nome e sexo. É, ainda, essencial saber qual a especialidade que o PS possui, pelo que todos estes atributos têm que ser não nulos, exceto o seu salário.

```

CREATE TABLE IF NOT EXISTS `centrosaude`.`Profissional_Saude` (
  `CC` INT NOT NULL,
  `Nome` VARCHAR(100) NOT NULL,
  `Sexo` VARCHAR(1) NOT NULL,
  `Salario` INT NULL,
  `Especialidade` INT NOT NULL,
  PRIMARY KEY (`CC`),
  INDEX `fk_Profissional_Saude_Especialidade1_idx` (`Especialidade` ASC) VISIBLE,
  CONSTRAINT `fk_Profissional_Saude_Especialidade1`
    FOREIGN KEY (`Especialidade`)
      REFERENCES `centrosaude`.`Especialidade` (`Id`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 16 - Código de criação tabela Profissional_Saude

5.2.6. Tabela Contacto_PS

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
PS	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>						
Contacto	VARCHAR(50)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>						

Figura 17 - Tabela que representa a entidade Contacto_PS

A tabela *Contacto_PS* é idêntica à tabela *Contacto_Utente*. Como o atributo *Contacto* é multivalue, foi criada uma tabela *Contacto_PS* cuja chave primária composta possui os identificadores do *PS* (*INT*) e do *Contacto* (*VARCHAR*).

```
CREATE TABLE IF NOT EXISTS `centrosaude`.`Contacto_PS` (
  `PS` INT NOT NULL,
  `Contacto` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`PS`, `Contacto`),
  CONSTRAINT `fk_Contacto_PS_Profissional_Saude1`
    FOREIGN KEY (`PS`)
    REFERENCES `centrosaude`.`Profissional_Saude` (`CC`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 18 - Código de criação tabela Contacto_PS

5.2.7. Tabela Especialidade

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>						
Descricao	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>						

Figura 19 - Tabela que representa a entidade Especialidade

Uma *Especialidade* é identificada por um *Id*, sendo a chave privada da tabela, não nulo e único. O atributo descrição é, também, não nulo, pois é necessário saber qual o nome da especialidade.

```
CREATE TABLE IF NOT EXISTS `centrosaude`.`Especialidade` (
  `Id` INT NOT NULL,
  `Descricao` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
```

Figura 20 - Código de criação tabela Especialidade

5.2.8. Tabela Consulta

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Id	INT	♦	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
HoraSaida	DATETIME	♦	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Descricao	VARCHAR(100)	♦	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Duracao	INT	♦	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Data	DATETIME	♦	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
HoraEntrada	DATETIME	♦	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Estado	INT	♦	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figura 21 - Tabela que representa a entidade Consulta

A uma consulta diz respeito um Id (chave primária, o que implica ser não nulo e único), uma hora de entrada e de saída, um estado e uma duração. Para que uma consulta seja marcada, é obrigatório que esta tenha uma *Data*, pelo que tem que ser necessariamente não nula.

Por fim, graças à existência de horas de entrada e de saída da consulta, é possível calcular a duração da consulta, inicialmente com valor nulo, com auxílio do *Trigger MeterDuracao*.

```
DELIMITER $
CREATE TRIGGER MeterDuracao BEFORE UPDATE ON Consulta
FOR EACH ROW
BEGIN
    IF new.HoraSaida IS NOT NULL THEN
        SET new.Duracao = TIMESTAMPDIFF(MINUTE,new.HoraEntrada,new.HoraSaida);
    END IF;
END $$;
DELIMITER ;
```

Figura 22 - Trigger meterDuracao

```
CREATE TABLE IF NOT EXISTS `centrosaude`.`Consulta` (
  `Id` INT NOT NULL,
  `HoraSaida` DATETIME NULL,
  `Descricao` VARCHAR(100) NULL,
  `Duracao` INT NULL,
  `Data` DATETIME NOT NULL,
  `HoraEntrada` DATETIME NULL,
  `Estado` INT NULL,
  PRIMARY KEY (`Id`)
)
ENGINE = InnoDB;
```

Figura 23 - Código criação tabela Consulta

5.2.9. Tabela PUC

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Utente	INT	✗	✓	✓	✗	✗	✗	✗	✗	
PS	INT	✗	✓	✓	✗	✗	✗	✗	✗	
Consulta	INT	✗	✓	✓	✗	✗	✗	✗	✗	
Preco	FLOAT	✗	✗	✓	✗	✗	✗	✗	✗	

Figura 24 - Tabela que representa a entidade PUC

Devido ao relacionamento ternário “N:N:N” entre as entidades *Profissional_Saude*, *Utente* e *Consulta*, foi criada uma tabela chamada *PUC*.

Esta tabela tem como chave primária composta as chaves primárias das restantes três (3) tabelas.

É, ainda, nesta tabela que se obtém o preço de cada consulta, cujo valor é definido de acordo com a idade dos utentes e se são, ou não, dadores de sangue. Para esse efeito, foi definido o *Trigger precoConsulta*.

```
DELIMITER $
CREATE TRIGGER precoConsulta BEFORE INSERT ON PUC
FOR EACH ROW
BEGIN
    IF ((SELECT FaixaEtaria FROM Utente WHERE Nr = new.Utente) = 'S') THEN SET new.Preco = 0;
    ELSE IF ((SELECT Dador FROM Utente WHERE Nr = new.Utente) = 1) THEN SET new.Preco = 4;
    ELSE SET new.Preco = 5;
    END IF;
END IF;
END $ 
DELIMITER ;
```

Figura 25 - Trigger precoConsulta

```
CREATE TABLE IF NOT EXISTS `centrosaude`.`PUC` (
  `Utente` INT NOT NULL,
  `PS` INT NOT NULL,
  `Consulta` INT NOT NULL,
  `Preco` FLOAT NOT NULL,
  PRIMARY KEY (`Utente`, `PS`, `Consulta`),
  INDEX `fk_PUC_Profissional_Saude1_idx`(`PS` ASC) VISIBLE,
  INDEX `fk_PUC_Consulta1_idx`(`Consulta` ASC) VISIBLE,
  CONSTRAINT `fk_PUC_Utente1`
    FOREIGN KEY (`Utente`)
    REFERENCES `centrosaude`.`Utente` (`Nr`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_PUC_Profissional_Saude1`
    FOREIGN KEY (`PS`)
    REFERENCES `centrosaude`.`Profissional_Saude` (`CC`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_PUC_Consulta1`
    FOREIGN KEY (`Consulta`)
    REFERENCES `centrosaude`.`Consulta` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 26 - Código criação tabela PUC

5.2.10. Tabela Receita

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Id	INT	◊	✓	✓	□	□	□	□	□	
Data	DATE	◊	□	✓	□	□	□	□	□	
Descricao	VARCHAR(100)	◊	□	□	□	□	□	□	□	
Consulta	INT	◊	□	✓	□	□	□	□	□	
...

Figura 27 - Tabela que representa a entidade Receita

A tabela *Receita* é composta por um *Id*, que corresponde à sua chave primária, uma data, não nula, e uma descrição. Tem como chave estrangeira o *Id* da *Consulta*, através do qual é relacionada com essa tabela, não podendo possuir um valor nulo.

```
CREATE TABLE IF NOT EXISTS `centrosaude`.`Receita` (
  `Id` INT NOT NULL,
  `Data` DATE NOT NULL,
  `Descricao` VARCHAR(100) NULL,
  `Consulta` INT NOT NULL,
  PRIMARY KEY (`Id`),
  INDEX `fk_Receita_Consulta1_idx`(`Consulta` ASC) VISIBLE,
  CONSTRAINT `fk_Receita_Consulta1`
    FOREIGN KEY (`Consulta`)
    REFERENCES `centrosaude`.`Consulta`(`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 28 - Código criação tabela Receita

5.2.11. Tabela Medicamento

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Id	INT	◊	✓	✓	□	□	□	□	□	
Nome	VARCHAR(45)	◊	□	✓	□	□	□	□	□	
Descricao	VARCHAR(45)	◊	□	□	□	□	□	□	□	

Figura 29 - Tabela que representa a entidade Medicamento

A tabela *Medicamento*, cuja chave primária é o *Id* (único e não nulo), é composta pelo nome do medicamento, sendo este, também, não nulo, e uma descrição.

```

CREATE TABLE IF NOT EXISTS `centrosaude`.`Medicamento` (
  `Id` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Descricao` VARCHAR(45) NULL,
  PRIMARY KEY (`Id`)
)
ENGINE = InnoDB;

```

Figura 30 - Código criação tabela Medicamento

5.2.12. Tabela Medicamento_Receita

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
Medicamento	INT	▼	✓	✓	□	□	□	□	□	
Receita	INT	▼	✓	✓	□	□	□	□	□	
Posologia	INT	▼	□	□	□	□	□	□	□	

Figura 31 - Tabela que representa a entidade Medicamento_Receita

Por fim, a tabela *Medicamento_Receita* tem como chave primária composta os *Ids* que da *Receita* e *Medicamento* em questão. Possui, ainda, a posologia, que indica o número de vezes que cada medicamento deve ser tomado por dia.

```

CREATE TABLE IF NOT EXISTS `centrosaude`.`Medicamento_Receita` (
  `Medicamento` INT NOT NULL,
  `Receita` INT NOT NULL,
  `Posologia` INT NULL,
  PRIMARY KEY (`Medicamento`, `Receita`),
  INDEX `fk_Medicamento_has_Receita_Receita1_idx` (`Receita` ASC) VISIBLE,
  INDEX `fk_Medicamento_has_Receita_Medicamento1_idx` (`Medicamento` ASC) VISIBLE,
  CONSTRAINT `fk_Medicamento_has_Receita_Medicamento1`
    FOREIGN KEY (`Medicamento`)
    REFERENCES `centrosaude`.`Medicamento` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Medicamento_has_Receita_Receita1`
    FOREIGN KEY (`Receita`)
    REFERENCES `centrosaude`.`Receita` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Figura 32 - Código criação tabela Medicamento_Receita

5.3. Tradução das interrogações do utilizador para SQL (alguns exemplos)

5.3.1. Funções

Ao longo do desenvolvimento desta BD, foram implementadas apenas duas funções, *bonusS* e *SemSobreposicao*. A primeira calcula qual o bónus que um PS vai ganhar ao final de um ano, com base no número de consultas que realizou por utente. Já a segunda, recebendo os identificadores do utente, do PS e a data de marcação, indica o número de consultas que se sobrepõem, quer a nível de um utente, quer a nível do profissional. No caso ideal de retornar o valor zero (0), é possível a marcação da consulta, não existindo colisões.

5.3.2. Procedimentos

De modo a tomar partido das funcionalidades que um SGBD oferece, foram criados vários procedimentos que ajudarão a manter a BD organizada e atualizada. Nesta secção, serão apresentados alguns exemplos de procedimentos criados que demonstrem a tradução das interrogações do utilizador para SQL.

- ObitoUtente

Este procedimento recebe como parâmetros o número do utente e a data do seu falecimento.

- ReceitasM

Ao inserir o identificador de um profissional de saúde, obtém-se uma lista com todas as receitas emitidas pelo mesmo, sendo possível visualizar o Id de cada receita e respetiva descrição.

- DinheiroMes

A fim de controlar a situação financeira do centro de saúde, foi criado este procedimento que obtém o capital adquirido durante determinado mês, mediante introdução do mês e ano requerido.

- ConsulXTemp

Este procedimento permite saber quantas consultas foram realizadas num determinado intervalo de tempo, recebendo como parâmetros as datas inicial e final desse intervalo.

- ReceitasEmConsulta

Durante uma consulta, um profissional de saúde pode emitir uma, ou mais, receitas. Desta forma, foi implementado este procedimento cujo objetivo é apresentar as receitas que

foram emitidas numa consulta para um utente. Assim, são dados como parâmetros o identificador do utente e a data em que se realizou a consulta em questão.

- recebeConsulta

Em semelhança com o procedimento *ReceitasM*, este procedimento apresenta uma lista de todas as consultas realizadas por um profissional de saúde, após ser dado como parâmetro o identificador do mesmo.

- bonus

Este procedimento visa apresentar o valor total do bónus que um profissional de saúde irá receber no final do ano. Após receber o Id do PS e o ano em questão, este procedimento vai ser executado juntamente com a função *bonusS*, sendo calculado e apresentado o valor do bónus.

- historicoClinico

O acesso ao histórico clínico dos utentes permite a continuidade de cuidados e o correto tratamento e encaminhamento dos mesmos atempadamente. Por este motivo, foi criado o procedimento que, fornecendo o número do utente, lista as anotações tomadas durante as consultas com as respetivas datas.

5.4. Tradução das transações estabelecidas para SQL (alguns exemplos)

Tendo em conta que ao efetuar uma transação é necessário garantir que a BD só será alterada caso nenhum erro seja detetado, têm de existir métodos que assegurem o mesmo. Dito isto, sempre que uma inserção, atualização ou remoção é feita sobre a *BD*, tem que ser garantida a consistência dos dados.

Tenham-se, a título de exemplo, as três (3) transações seguintes.

5.4.1. Marcação de uma Consulta

Recebendo o número do utente, o número do cartão de cidadão do PS que efetuará a consulta, uma data de marcação e o respetivo identificador da consulta, esta será marcada, ou seja, inserida na BD, ficando com o atributo de estado a um (1), índice da realização da marcação. Caso o utente tenha falecido, é enviado um sinal a notificar e a consulta não é marcada. Não pode ser esquecida a existência de uma função que ajuda a não permitir a sobreposição de consultas.

```
-- Implementação de um procedure para marcar consultas
DELIMITER $$ 
CREATE PROCEDURE MarcarConsulta(NrUtente INT, NrPS INT, NrConsulta INT, Data DATETIME)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;

    IF ((SELECT DtObito FROM Utente WHERE Nr = NrUtente) IS NOT NULL) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Não é possível agendar a consulta - Utente faleceu';
    ELSE
        IF (SELECT SemSobreposicao(NrUtente, NrPS, Data) > 0) THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Não é possível agendar a consulta - Sobreposição de consultas';
        ELSE
            INSERT INTO Consulta (Id, Data, Estado) VALUES (NrConsulta, Data, 1);
            INSERT INTO PUC (Utente, PS, Consulta) VALUES (NrUtente, NrPS, NrConsulta);
        END IF;
    END IF;

    IF erro = 1
    THEN BEGIN
        ROLLBACK;
        SELECT 'Possibilidades: Utente faleceu; existe sobreposição de consultas; horário de agendamento não é possível';
    END;
    ELSE COMMIT;
    END IF;
END $$ 
DELIMITER ;
```

Figura 33 - Procedure MarcarConsulta

5.4.2. Registo de um PS

De modo a adicionar um profissional de saúde à BD, é necessário inserir o registo em estudo na tabela *Profissional_Saude*. Este terá o seu número de cartão de cidadão, nome, sexo, salário e identificação de qual a sua especialidade.

```
-- Implementação de um procedure que insere um PS na BD
DELIMITER $$ 
CREATE PROCEDURE InserirPS(CC INT, Nome VARCHAR(100), Sexo VARCHAR(1), Salario INT, Especialidade INT)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;
    INSERT INTO Profissional_Saude VALUES (CC, Nome, Sexo, Salario, Especialidade);

    IF erro = 1
    THEN
        BEGIN
        ROLLBACK;
        SELECT 'Sexo indicado pode não ser válido';
        END;
    ELSE COMMIT;
    END IF;
END $$ 
DELIMITER ;
```

Figura 34 - Procedure InserirPS

5.4.3. Remarcar uma Consulta

De modo a garantir que a remarcação da consulta é realizada corretamente, a consulta anteriormente marcada tem de ver o seu estado alterado para “quatro” (4), valor que indica o cancelamento. Como um utente apenas pode recorrer a uma única consulta a uma determinada hora, apenas é necessário saber qual a data e hora de marcação e o seu número na instituição. Em seguida, terá que ser marcada uma nova data, parâmetro recebido no *input*, bem como o identificador do PS e da nova consulta a ser marcada.

```

-- Implementação de um procedure que permite remarcar uma consulta : Cancela a antiga e cria uma nova
DELIMITER $$ 
CREATE PROCEDURE RemarcarConsulta(NrUtente INT, data_velha DATETIME, data_nova DATETIME, NrPS INT, NrConsulta INT)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;

    CALL DesmarcarConsulta(NrUtente, data_velha);
    CALL MarcarConsulta(NrUtente, NrPS, NrConsulta, data_nova);

    IF erro = 1
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END $$ 
DELIMITER ;

```

Figura 35 - Procedure RemarcarConsulta

5.5. Escolha, definição e caracterização de índices em SQL

Os índices são utilizados, na sua maioria, para tornar as operações de procura de dados das tabelas mais rápidas, sendo claramente úteis quando se tratam de tabelas de grandes dimensões. Permitem uma rápida determinação da posição dos dados indexados, não percorrendo sequencialmente as tabelas, resultando num decréscimo do número de acessos à memória e do tempo de procura.

O uso de índices na BD apenas foi realizado no que toca às chaves primárias e estrangeiras. Como InnoDB, o mecanismo de armazenamento para o MySQL, requer a indexação das chaves primárias e estrangeiras, de modo a que não sejam necessárias pesquisas a fundo em cada tabela, aquando da procura de uma chave, deve existir um índice onde as chaves estão listadas. Este índice é criado, automaticamente, numa tabela de referência, caso não exista.

Assim, devido à, atual, dimensão reduzida da *BD* e, tendo em conta os pedidos dos utilizadores, apenas foram indexadas as chaves primárias e estrangeiras.

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

A dimensão da BD, após a sua definição, apenas tem em consideração a estrutura da mesma, os *Triggers* e funções necessárias para manter a integridade e consistência dos dados. Sendo assim, sem qualquer tipo de dados inseridos nas tabelas, o valor inicial obtido para o base de dados foi de 320.0 *KiB*.

Com o objetivo de prever o espaço em disco ocupado pela base de dados segundo uma estimativa do crescimento previsto, calculou-se o tamanho de cada registo de cada uma das tabelas. Para tal, tendo em conta o tamanho de cada tipo de dados, segundo a documentação do MySQL [5], foi obtida a Tabela 13, com o tamanho médio de cada registo.

Tabela	Tamanho médio ocupado por registo (Bytes)
Utente	222
Profissional_Saude	113
Consulta	136
Receita	111
Medicamento	94
Especialidade	54
Contacto_PS	54
Contacto_Utente	54
Incapacidade	104
Utente_Incapacidade	8
PUC	16
Medicamento_Receita	12

Tabela 13 - Tamanho médio ocupado por cada registo de cada tabela da base de dados

Finito o povoamento inicial da BD, a quantidade de cada registo por tabela foi representada na Tabela 14.

Profissional_Saude	6
Consulta	32
Receita	13
Medicamento	13
Especialidade	5
Contacto_PS	6
Contato_Utente	6
Incapacidade	12
Utente_Incapacitado	12
PUC	32
Medicamento_Receita	14

Tabela 14 - Quantidade de registos por tabela

Sendo assim, o tamanho em disco após este povoamento será de:

$$20 * 222 + 6 * 113 + 32 * 136 + 13 * 111 + 13 * 94 + 5 * 54 + 6 * 54 + 6 * 54 + 12 * 104 + 12 * 8 + 32 * 16 + 14 * 12 = \\ = 15077 \text{ Bytes}$$

Deste modo, a BD possui aproximadamente 15KB após o primeiro povoamento.

Relativamente ao crescimento futuro, é expectável que, diariamente, haja mais consultas a ser marcadas, tal como mais utentes a visitar o centro de saúde, até que a grande maioria esteja registada. Não se espera uma mudança nos profissionais de saúde num futuro próximo, pelo que esse campo se manterá inalterado. Assumindo que o número máximo de consultas a ser efetuadas diariamente é quarenta (40), tendo em conta o tempo médio de consulta e descontando eventuais e inevitáveis atrasos, sabendo que existem dois (2) dias em que a unidade se encontra encerrada, serão possíveis efetuar catorze mil e quinhentas e vinte (14520) consultas num ano. Cada consulta gera, em média, uma receita com um medicamento associado. Os medicamentos receitados deverão permanecer pouco alterados, devido à homogeneidade dos motivos nas idas dos utentes à unidade. As especialidades necessárias

para o bom funcionamento da instituição também deverão permanecer inalteradas. Assume-se, ainda, que apenas um terço (1/3) dos utentes em média possui algum tipo de incapacidade. O número de utentes permanecerá relativamente constante a partir do momento em que todos se encontrem registados, uma vez que o falecimento de alguns será compensado pelo nascimento de outros.

Assuma-se que a freguesia onde se encontra este centro de saúde é povoada por dois mil (2000) habitantes. Ao final de um ano será expectável que a *BD* tenha uma dimensão de:

$$2000 * 222 + 6 * 113 + 14520 * 136 + 14520 * 111 + 13 * 94 + 5 * 54 + 6 * 54 + 2000 * 54 + 12 * 104 + 667 * 8 + 14520 * 16 + 14520 * 12 = 4554078 \text{ Bytes}$$

Sendo assim, um ano completamente preenchido de consultas implica um aumento do espaço em disco de, aproximadamente, 4.554078MB.

5.7. Definição e caracterização das vistas de utilização em SQL

Uma vista é um objeto de dados que não contém dado algum, sendo que o conteúdo de uma vista é resultante de uma tabela de base. As operações sobre estas funcionam da mesma forma que as operações sobre tabelas, só que não contêm dados próprios. O maior diferenciador entre uma vista e uma tabela é que uma vista é construída sobre uma ou várias tabelas e/ou vistas. Caso os dados sejam alterados na tabela subjacente, a alteração será refletida na vista.

Ao longo do desenvolvimento do modelo físico foi identificado um utilizador da futura BD, o médico. De seguida vão ser apresentadas as vistas dos médicos sobre a BD.

5.7.1. Vista Medicamento

A seguinte vista permite ao médico consultar os medicamentos disponíveis para subscrição, juntamente com a descrição dos mesmos.

```
DROP VIEW IF EXISTS vwMedicamentos;
CREATE VIEW vwMedicamentos AS
    SELECT DISTINCT Id AS 'Identificação',
        Nome AS 'Nome',
        Descricao AS 'Informação Extra'
    FROM Medicamento;
```

Figura 36 - Vista vwMedicamentos

5.7.2. Vista Utente

A vista Utente permite ao médico consultar todos os utentes inscritos na unidade de saúde, juntamente com informação como o seu nome, a sua faixa etária, a informação se o utente é dador de sangue e também o sexo do mesmo.

```
DROP VIEW IF EXISTS vwUtentes;
CREATE VIEW vwUtentes AS
    SELECT DISTINCT Nr AS 'Identificação',
        Nome AS 'Nome',
        FaixaEtaria AS 'Faixa Etária',
        Dador AS 'Dador',
        Sexo AS 'Sexo'
    From Utente
    WHERE DtObito IS NULL;
```

Figura 37 - Vista vwUtentes

5.7.3. Vista óbito

A seguinte vista permite ao médico consultar todos os utentes inscritos na unidade de saúde que faleceram, juntamente com informação sobre a data de óbito e o número de utente do falecido.

```
DROP VIEW IF EXISTS listaObito;
CREATE VIEW listaObito AS
    SELECT U.Nome, U.Nr, U.DtObito
    FROM Utente U
    WHERE U.DtObito IS NOT NULL
    ORDER BY U.Nome ASC;
```

Figura 38 - Vista listaObito

5.8. Definição e caracterização dos mecanismos de segurança em SQL

A BD desenvolvida neste projeto suporta apenas um utilizador, além do administrador, que é o médico. Foi então analisado pelo grupo quais as tabelas e operações a que estes podem ter acesso.

Os PS têm então a capacidade de prescrever receitas, marcar, desmarcar e remarcar consultas, consultar o histórico clínico dos seus pacientes, obter a listagem de medicamentos e dos utentes da unidade de saúde, estejam esses vivos ou não.

Na tabela 15 estão representadas as permissões possuídas pelo PS.

	PS			
	Select	Insert	Update	Delete
EmitirReceita	X	X		
MarcarConsulta	X	X		
DesmarcarConsulta	X		X	
RemarcarConsulta	X	X	X	
HistoricoClinico	X			
ListaObito	X			
vwUtentes	X			
vwMedicamentos	X			

Tabela 15 - Permissões de um profissional de saúde na BD

Em seguida mostra-se o código de criação do utilizador Médico, bem como as suas permissões.

```
DROP USER IF EXISTS 'Medico'@'localhost';
CREATE USER 'Medico'@'localhost' IDENTIFIED BY '12345';
GRANT EXECUTE ON PROCEDURE centrosaude.EmitirReceita TO 'Medico'@'localhost';
GRANT EXECUTE ON PROCEDURE centrosaude.MarcarConsulta TO 'Medico'@'localhost';
GRANT EXECUTE ON PROCEDURE centrosaude.DesmarcarConsulta TO 'Medico'@'localhost';
GRANT EXECUTE ON PROCEDURE centrosaude.RemarcarConsulta TO 'Medico'@'localhost';
GRANT EXECUTE ON PROCEDURE centrosaude.historicoClinico TO 'Medico'@'localhost';
GRANT SELECT ON centrosaude.listaObito TO 'Medico'@'localhost';
GRANT SELECT ON centrosaude.vwUtentes TO 'Medico'@'localhost';
GRANT SELECT ON centrosaude.vwMedicamentos TO 'Medico'@'localhost';
FLUSH PRIVILEGES;
SHOW GRANTS FOR 'Medico'@'localhost' ;
```

Figura 39 - Código de criação utilizador Médico

5.9. Revisão do sistema implementado com o utilizador

Após concluído o desenvolvimento da BD, foi necessário reunir com os utilizadores do sistema para estes o avaliarem, uma última vez.

Foi então marcada uma reunião com os representantes da unidade hospitalar, onde lhes foi apresentada uma demonstração de todas as funcionalidades implementadas, com vista à obtenção da aprovação por parte destes.

Depois de concluída a sessão de apresentação, os representantes da unidade hospitalar confirmaram que o sistema funcionava como pretendido, dando assim, fim ao desenvolvimento do projeto.

6. NoSQL

6.1. Justificação da utilização de um sistema NoSQL

Com um sistema de bases de dados relacional, é utilizada a linguagem SQL, cujas funcionalidades essenciais são as de definição (DDL) e manipulação (DML) de dados. Num sistema de bases de dados relacional, existe a possibilidade de recuperação face a possíveis falhas, permitindo garantir a consistência dos dados, seguindo o modelo ACID (*Atomicidade, Consistência, Isolamento, Durabilidade*) para preservar a integridade de uma transação, pelo que os modelos relacionais são bastante consistentes.

No entanto, apresenta uma estrutura e tipo de dados fixos, o que significa que o armazenamento de informações sobre um novo dado implica a alteração da base de dados, durante a qual esta deve ser colocada offline. Possui uma escalabilidade vertical, pelo que a um único servidor devem ser adicionados recursos.

Por outro lado, um sistema “*Not Only SQL*” (NoSQL) adota um modelo não relacional, que se caracteriza por ser mais flexível relativamente às propriedades ACID.

Desta forma, um sistema NoSQL possui escalabilidade horizontal, que consiste em aumentar o número de máquinas disponíveis, permitindo a distribuição dos dados pelas várias máquinas. A ausência/flexibilidade do esquema permite responder a requisitos de alta escalabilidade necessários para gerir grandes quantidades de dados, bem como garantir a alta disponibilidade dos mesmos. Esta característica deve-se às propriedades BASE (*Basicamente Disponível, Estado Leve, Eventualmente Consistente*) dos sistemas NoSQL, que se focam em assegurar a disponibilidade dos dados, não garantindo a integridade dos mesmos. Assim, identifica-se a característica de nem sempre ser consistente, que tem como princípio o teorema CAP (*Consistência, Disponibilidade, Tolerância ao Particionamento*), que afirma que não é possível garantir mais do que duas das três propriedades apresentadas. Por fim, um sistema NoSQL possui suporte a replicação, que permite a replicação de forma nativa, aumentando a escalabilidade e diminuindo o tempo gasto para a recuperação de informação, bem como uma API simples, permitindo que qualquer aplicação tenha acesso aos dados de forma rápida e eficaz. [6] [7]

Prevendo o crescimento futuro do centro de saúde, que implica a adição de novos Profissionais de Saúde e Utentes, bem como a constante adição de consultas, a utilização de um sistema NoSQL demonstra ser a mais adequada, considerando as características previamente referidas.

6.2. Identificação e descrição dos objetivos da base de dados

A base de dados, agora implementada com um sistema NoSQL (MongoDB), visa possibilitar uma maior facilidade de agendamento e controlo de consultas, bem como aceder às informações referentes a cada utente e profissional de saúde, de forma mais rápida e de modo a suportar a introdução de dados em grande escala.

6.3. Identificação e explicação do tipo de questões que serão realizadas sobre o sistema de dados NoSQL

- Questão 1: Apresentar todas as consultas realizadas, marcadas e canceladas;
- Questão 2: Apresentar a informação de todos os utentes do centro de saúde;
- Questão 3: Apresentar a informação de todos os profissionais de saúde;
- Questão 4: Apresentar a informação de todos os medicamentos;
- Questão 5: Apresentar todas as pessoas que já faleceram;
- Questão 6: Verificar quais as receitas emitidas por determinado Profissional de Saúde;

Esta questão foi realizada de forma a facilitar a emissão de receitas, tendo por base as receitas previamente emitidas, bem como de controlar o número e tipo de receitas emitidas. Deste modo, recorre-se à coleção das consultas filtrando aquelas realizadas pelo profissional de saúde em questão, e apresentando todas as receitas por ele emitidas.

- Questão 7: Verificar quais as consultas associadas a um determinado Profissional de Saúde;

A possibilidade de aceder à lista de consultas que um profissional de saúde realizou ou irá realizar é crucial para o prévio conhecimento por parte do mesmo, e também para controlo do número de consultas por parte dos administrativos. Desta forma, recorre-se à coleção das consultas, filtrando aquelas realizadas pelo profissional de saúde em questão.

- Questão 8: Apresentar o histórico clínico de um dado utente.

Dado que vários profissionais de saúde podem realizar consultas sobre o mesmo utente, é indispensável que estes possam ter acesso ao histórico clínico do utente. Assim, recorre-se à coleção das consultas, filtrando aquelas que se encontram realizadas (Estado = 3) para o dado utente, apresentando o id da consulta em que o estado clínico foi escrito, bem como a descrição do mesmo e a data em que essa consulta se deu.

Estas questões foram adicionadas, uma vez que é de extrema relevância aceder constantemente aos dados referidos nas mesmas e, por este motivo, o acesso a estes tornava-se lento no panorama relacional.

6.4. Estrutura base para o sistema de dados NoSQL

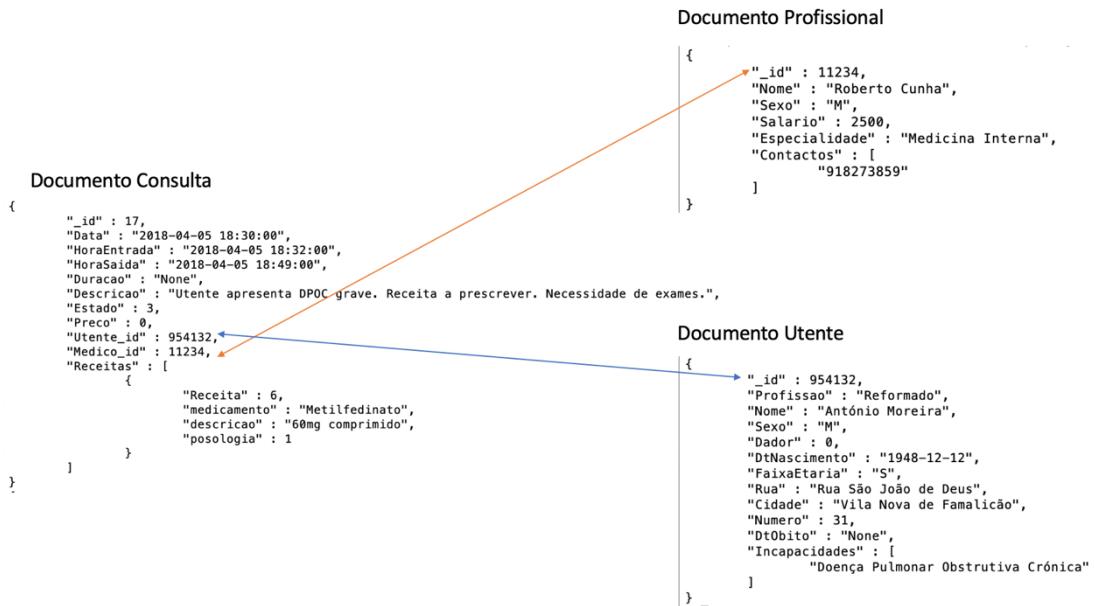


Figura 40 - Estrutura base para o sistema de dados NoSQL.

Como se pode observar na Figura 40, a estrutura base que foi definida para representar a BD em NoSQL é composta por três coleções: consultas, profissionais e utentes. Um documento *Consulta* é constituído por um Id, pela Data em que foi realizada, o horário da mesma, uma descrição, um Estado, um Preço e também pelas Receitas emitidas na mesma. Este documento relaciona-se com os outros dois documentos através do `Medico_id` e `Utente_id`.

Já um documento *Profisional* possui um Id, Nome, Sexo, Salário, Especialidade e Contactos.

Por fim, um documento *Utente* é constituído por um Id, Profissão, Nome, Sexo, um identificador para se o mesmo é dador de órgãos ou não, uma Data de Nascimento, Faixa Etária, Morada, uma Data de Óbito e por fim as suas Incapacidades.

A relação entre estes documentos cria uma BD NoSQL que representa o Estabelecimento de Saúde que o grupo se propôs a modelar.

6.5. Identificação dos objetos de dados no sistema SQL que serão utilizados para alimentar o novo sistema

Os objetos de dados do sistema SQL que serão utilizados para alimentar o novo sistema são as tabelas que descrevem as entidades do sistema e as tabelas que representam os relacionamentos entre estas.

As tabelas necessárias para a alimentação do novo sistema são as seguintes:

- Consulta;
- Receita;
- Medicamento_Receita;
- Medicamento;
- Profissional_Saude;
- Especialidade;
- Utente;
- Incapacidade;
- Utente_Incapacidade;
- PUC.

6.6. Processo de migração de dados

Tendo em conta a sugestão do professor regente da UC, o método escolhido foi codificar um *script* em Python, lendo cada elemento da BD relacional, inserindo-o na não relacional. Assim, sabendo que MongoDB não suporta *joins*, o esquema será diferente do esquema relacional, existindo algumas alterações na forma como os dados são guardados.

6.7. Explicação do processo de migração de dados

O primeiro passo a executar é a extração dos dados existentes na BD relacional. Para isto, foi necessário executar *queries* sobre os dados, na forma de *selects*, de modo a ser possível correr todas as linhas das tabelas, uma a uma.

Em seguida, teve que ser executada uma transformação dos tipos de dados para os tipos correspondentes, que existiam em MongoDB, como foi o caso do tipo *DATE* e *DATETIME* de MySQL, não compatíveis com o sistema não relacional.

Por fim, é necessário carregar os dados para o sistema de gestão de BD não relacional, MongoDB, através da execução de *queries* de inserção e *update*. Todo este processo foi realizado com auxílio das bibliotecas fornecidas pela linguagem de programação Python, permitindo uma fácil realização da tarefa pedida.

6.8. Implementação do processo de migração de dados

Começou-se por executar uma *query* que devolvesse todos os utentes, de modo a um a um, inseri-los no sistema de bases de dados não relacional. Em seguida, foi necessário tratar dos relacionamentos que existiam com a entidade utente, nomeadamente os seus contactos e incapacidades. Deste modo, são executadas duas *queries* para obter os campos mencionados. Estes serão, depois, inseridos na coleção utentes. Assim, cada utente fica com um formato semelhante à Figura 41.

```
{  
    "_id" : 137142,  
    "Profissao" : "Professor",  
    "Nome" : "Rodolfina Oliveira",  
    "Sexo" : "F",  
    "Dador" : 1,  
    "DtNascimento" : "1984-03-05",  
    "FaixaEtaria" : "A",  
    "Rua" : "Rua da Porta",  
    "Cidade" : "Caldas de Vizela",  
    "Numero" : 1,  
    "DtObito" : "None",  
    "Contactos" : [  
        "919187762"  
    ],  
    "Incapacidades" : [  
        "Toxicodependência",  
        "Obesidade"  
    ]  
}
```

Figura 41 - Exemplo de um documento na coleção Utente.

Em seguida, foi necessário fazer a migração dos profissionais de saúde. Utilizando um método semelhante ao dos utentes, tirando o facto de os atributos serem diferentes e de apenas possuir uma especialidade.

```
{  
    "_id" : 78352,  
    "Nome" : "Feliciano Freitas",  
    "Sexo" : "F",  
    "Salario" : 2200,  
    "Especialidade" : "Medicina Geral e Familiar",  
    "Contactos" : [  
        "912346132"  
    ]  
}
```

Figura 42 - Exemplo de um documento na coleção Profissionais.

Por fim, foi efetuada a migração das consultas. Estas possuem um identificador que, tal como nos casos de utentes e profissionais, foi aproveitado do anteriormente atribuído na BD relacional, uma data de marcação, respetiva hora de entrada e saída, com consequente duração, uma descrição, um estado, um preço e os identificadores relativos ao utente e PS, de forma a não existir incluindo as suas receitas e medicamentos, na forma de documentos embebidos.

```
{
    "_id" : 17,
    "Data" : "2018-04-05 18:30:00",
    "HoraEntrada" : "2018-04-05 18:32:00",
    "HoraSaida" : "2018-04-05 18:49:00",
    "Duracao" : "None",
    "Descricao" : "Utente apresenta DPOC grave. Receita a prescrever. Necessidade de exames.",
    "Estado" : 3,
    "Preco" : 0,
    "Utente_id" : 954132,
    "Medico_id" : 11234,
    "Receitas" : [
        {
            "Receita" : 6,
            "medicamento" : "Metilfedinato",
            "descricao" : "60mg comprimido",
            "posologia" : 1
        }
    ]
}
```

Figura 43 - Exemplo de um documento na coleção Consultas.

O *script* utilizado para a obtenção da migração dos dados encontra-se em anexo.

6.9. Apresentar a forma como as questões identificadas previamente podem ser satisfeitas com o novo sistema

- Questão 1

```
// Apresentar todas as consultas realizadas, marcadas e canceladas  
db.consultas.find().pretty();
```

Figura 44 - Questão 1.

- Questão 2

```
// Apresentar a informação de todos os utentes do centro de saúde  
db.utentes.find().pretty();
```

Figura 45 - Questão 2.

- Questão 3

```
// Apresentar a informação de todos os profissionais de saúde  
db.profissionais.find().pretty();
```

Figura 46 - Questão 3.

- Questão 4

```
// Apresentar a informação de todos os medicamentos  
db.consultas.distinct( "Receitas.medicamento")
```

Figura 47 - Questão 4.

- Questão 5

```
// Apresentar todas as pessoas que já faleceram  
db.utentes.find({ "DtObito":{$ne:"None"}}, { "Nome" : 1, "Nr" : 1, "DtObito": 1});
```

Figura 48 - Questão 5.

- Questão 6

```
// Verificar quais as receitas emitidas por determinado Profissional de Saúde  
db.consultas.distinct("Receitas.Receita",{"Medico_id" : ....})
```

Figura 49 - Questão 6

- Questão 7

```
// Verificar quais as consultas associadas a um determinado Profissional de Saúde  
db.consultas.find({ "Medico_id" : ....}).pretty()
```

Figura 50 - Questão 7.

- Questão 8

```
// Apresentar o histórico clínico de um dado utente  
db.consultas.find({ "Utente_id" : {$eq:...}, "Estado" : {$eq : 3}}, {"_id" : 1, "Descricao" : 1, "Data" : 1}).pretty()
```

Figura 51 - Questão 8.

7. Conclusões e Trabalho Futuro

O desenvolvimento desta BD permitiu comprovar o aumento de eficiência obtido com a sua implementação, tanto nos processos de organização, como nos processos de manutenção do que é pretendido modelar.

O modelo de gestão da Unidade Hospitalar, desenvolvido ao longo do projeto, viabiliza a facilidade no acesso e gestão dos dados associados à Unidade Hospitalar, permitindo agendar e realizar consultas, alterando, automaticamente, o estado da consulta quando necessário. Possibilita, ainda, o acesso ao histórico clínico do doente, facilitando o acompanhamento do mesmo. De forma a aumentar a eficiência da Unidade Hospitalar, foi efetuada uma otimização na marcação de consultas, não permitindo sobreposições nas mesmas.

Apesar das mais valias implementadas, seria interessante, no futuro, o desenvolvimento de uma interface gráfica de modo a que a visualização do funcionamento da BD se torne mais apelativa e intuitiva. Seria, também, positivo o aumento da complexidade em termos do número de entidades existentes na BD, criando assim um modelo mais próximo da realidade de uma Unidade Hospitalar. Outro aspeto a melhorar seria definir um comando que permitisse atualizar a faixa etária de cada utente à medida que este envelhece.

No que diz respeito à implementação da BD num panorama não relacional, foi comprovado um aumento da eficiência na execução de interrogações simples, como seria esperado, de modo a permitir várias execuções, sem um período de espera elevado. Assim, o desenvolvimento de ambos os tipos de BDs permitiu uma melhor noção das vantagens e desvantagens de cada um.

Numa perspetiva futura, com a ideia de melhorar este projeto, seria proveitoso implementar as funcionalidades referidas anteriormente, permitindo não só um aumento da complexidade interna da BD, mas também uma simplificação no que toca à utilização por parte dos usuários.

8. Referências

- [1] Chedv.min-saude.pt. (2018). “Centro de Saúde”. [online] Available at: http://www.chedv.min-saude.pt/index.php?option=com_content&view=article&id=106:2-centro-de-saude&catid=39:guia-online-do-utente&Itemid=201 [Accessed 13 Nov. 2018].
- [2] Sns.gov.pt. (2018). Programas de Saúde Prioritários. [online] Available at: <https://www.sns.gov.pt/institucional/programas-de-saude-prioritarios/> [Accessed 13 Nov. 2018].
- [3] Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management, Addison-Wesley, 4^a Edição, 2004.
- [4] Dgs.pt. (2018). *Plano Nacional de Saúde » Metas globais para o sistema de saúde*. [online] Available at: <https://www.dgs.pt/pns-indicadores-e-metas/sistema-de-saude.aspx> [Accessed 17 Nov. 2018].
- [5] Data Type Storage Requirements, MySQL 8.0 Reference Manual. [online] Available at: <https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html>
- [6] MongoDB.(2019). What is NoSQL?. [online] Available at: www.mongodb.com/nosql-inline [Accessed 15 Jan. 2019].
- [7] MongoDB. (2019). NoSQL Databases Explained.[online] Available at: <https://www.mongodb.com/nosql-explained> [Accessed 15 Jan. 2019].

Lista de Siglas e Acrónimos

BD Base de Dados

SQL Structured Query Language

OLTP On-Line Analytical Processing

PS Profissional de Saúde

SGBD Sistema Gestão Base de Dados

SNS Serviço Nacional de Saúde

DGS Direção Geral da Saúde

Id Identificação

ML Modelo Lógico

CC Cartão de Cidadão

DDL Data Definition Language

DML Data Manipulation Language

ACID Atomicidade, Consistência, Isolamento, Durabilidade

NoSQL Not Only Structured Query Language

CAP Consistência, Disponibilidade, Tolerância ao Particionamento

API Application Programming Interface

BASE Basicamente Disponível, Estado leve, Eventualmente disponível

Anexos

Nesta secção encontra-se informação adicional necessária para melhor compreensão do projeto realizado.

Anexo I – Código DDL

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,
ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----
-- Schema centrosaude
-- -----


CREATE SCHEMA IF NOT EXISTS `centrosaude` DEFAULT CHARACTER SET utf8 ;
USE `centrosaude` ;

-- -----
-- Table `centrosaude`.`Incapacidade`
-- -----


CREATE TABLE IF NOT EXISTS `centrosaude`.`Incapacidade` (
  `Id` INT NOT NULL,
  `Descricao` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Utente`
-- -----


CREATE TABLE IF NOT EXISTS `centrosaude`.`Utente` (
  `Nr` INT NOT NULL,
  `Profissao` VARCHAR(30) NULL,
  `Nome` VARCHAR(75) NOT NULL,
  `Sexo` VARCHAR(1) NOT NULL,
  `Dador` TINYINT NULL,
  `DtNascimento` DATE NOT NULL,
  `FaixaEtaria` VARCHAR(1) NULL,
  `Rua` VARCHAR(50) NULL,
  `Cidade` VARCHAR(50) NULL,
  `Numero` INT NULL,
  `DtObito` DATE NULL,
```

```

        PRIMARY KEY (`Nr`)
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Contacto_Utente`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Contacto_Utente` (
    `Utente` INT NOT NULL,
    `Contacto` VARCHAR(50) NOT NULL,
    PRIMARY KEY (`Utente`, `Contacto`),
    CONSTRAINT `fk_Contacto_Utente_Utente`
        FOREIGN KEY (`Utente`)
            REFERENCES `centrosaude`.`Utente` (`Nr`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Utente_Incapacitado`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Utente_Incapacitado` (
    `Incapacidade` INT NOT NULL,
    `Utente` INT NOT NULL,
    PRIMARY KEY (`Incapacidade`, `Utente`),
    INDEX `fk_Incapacidade_has_Utente_Utente1_idx` (`Utente` ASC) VISIBLE,
    INDEX `fk_Incapacidade_has_Utente_Incapacidade1_idx` (`Incapacidade` ASC)
VISIBLE,
    CONSTRAINT `fk_Incapacidade_has_Utente_Incapacidade1`
        FOREIGN KEY (`Incapacidade`)
            REFERENCES `centrosaude`.`Incapacidade` (`Id`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_Incapacidade_has_Utente_Utente1`
        FOREIGN KEY (`Utente`)
            REFERENCES `centrosaude`.`Utente` (`Nr`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Especialidade`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Especialidade` (

```

```

`Id` INT NOT NULL,
`Descricao` VARCHAR(50) NOT NULL,
PRIMARY KEY (`Id`))
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Profissional_Saude`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Profissional_Saude` (
`CC` INT NOT NULL,
`Nome` VARCHAR(100) NOT NULL,
`Sexo` VARCHAR(1) NOT NULL,
`Salario` INT NULL,
`Especialidade` INT NOT NULL,
PRIMARY KEY (`CC`),
INDEX `fk_Profissional_Saude_Especialidade1_idx`(`Especialidade` ASC)
VISIBLE,
CONSTRAINT `fk_Profissional_Saude_Especialidade1`
FOREIGN KEY (`Especialidade`)
REFERENCES `centrosaude`.`Especialidade`(`Id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Contacto_PS`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Contacto_PS` (
`PS` INT NOT NULL,
`Contacto` VARCHAR(50) NOT NULL,
PRIMARY KEY (`PS`, `Contacto`),
CONSTRAINT `fk_Contacto_PS_Profissional_Saude1`
FOREIGN KEY (`PS`)
REFERENCES `centrosaude`.`Profissional_Saude`(`CC`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Consulta`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Consulta` (
`Id` INT NOT NULL,

```

```

`HoraSaida` DATETIME NULL,
`Descricao` VARCHAR(100) NULL,
`Duracao` INT NULL,
`Data` DATETIME NOT NULL,
`HoraEntrada` DATETIME NULL,
`Estado` INT NULL,
PRIMARY KEY (`Id`))
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Receita`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Receita` (
`Id` INT NOT NULL,
`Data` DATE NOT NULL,
`Descricao` VARCHAR(100) NULL,
`Consulta` INT NOT NULL,
PRIMARY KEY (`Id`),
INDEX `fk_Receita_Consulta1_idx` (`Consulta` ASC) VISIBLE,
CONSTRAINT `fk_Receita_Consulta1`
    FOREIGN KEY (`Consulta`)
    REFERENCES `centrosaude`.`Consulta` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Medicamento`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Medicamento` (
`Id` INT NOT NULL,
`Nome` VARCHAR(45) NOT NULL,
`Descricao` VARCHAR(45) NULL,
PRIMARY KEY (`Id`))
ENGINE = InnoDB;

-- -----
-- Table `centrosaude`.`Medicamento_Receita`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`Medicamento_Receita` (
`Medicamento` INT NOT NULL,
`Receita` INT NOT NULL,
`Posologia` INT NULL,

```

```

PRIMARY KEY (`Medicamento`, `Receita`),
INDEX `fk_Medicamento_has_Receita_Receita1_idx`(`Receita` ASC) VISIBLE,
INDEX `fk_Medicamento_has_Receita_Medicamento1_idx`(`Medicamento` ASC)
VISIBLE,
CONSTRAINT `fk_Medicamento_has_Receita_Medicamento1`
FOREIGN KEY (`Medicamento`)
REFERENCES `centrosaude`.`Medicamento`(`Id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_Medicamento_has_Receita_Receita1`
FOREIGN KEY (`Receita`)
REFERENCES `centrosaude`.`Receita`(`Id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `centrosaude`.`PUC`
-- -----
CREATE TABLE IF NOT EXISTS `centrosaude`.`PUC` (
`Utente` INT NOT NULL,
`PS` INT NOT NULL,
`Consulta` INT NOT NULL,
`Preco` FLOAT NOT NULL,
PRIMARY KEY (`Utente`, `PS`, `Consulta`),
INDEX `fk_PUC_Profissional_Saude1_idx`(`PS` ASC) VISIBLE,
INDEX `fk_PUC_Consulta1_idx`(`Consulta` ASC) VISIBLE,
CONSTRAINT `fk_PUC_Utente1`
FOREIGN KEY (`Utente`)
REFERENCES `centrosaude`.`Utente`(`Nr`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_PUC_Profissional_Saude1`
FOREIGN KEY (`PS`)
REFERENCES `centrosaude`.`Profissional_Saude`(`CC`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_PUC_Consulta1`
FOREIGN KEY (`Consulta`)
REFERENCES `centrosaude`.`Consulta`(`Id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

Anexo II – Código do Povoamento Inicial

```

USE `centrosaude`;
-- povoamento Especialidade
INSERT INTO Especialidade (Id, Descricao) VALUES
(1, 'Medicina Interna'),
(2, 'Nutrição'),
(3, 'Psiquiatria'),
(4, 'Psicologia'),
(5, 'Medicina Geral e Familiar');

-- povoamento Profissional_Saude
INSERT INTO Profissional_Saude (Nome, CC, Sexo, Salario, Especialidade) VALUES
('Roberto Cunha',11234,'M', 2500,1),
('Anastácia Pereira',98207,'F', 2000,2),
('Diogo Magalhães',19981,'M', 1400,3),
('Maria Pinho',56277,'F', 2900,4),
('Felicia Freitas',78352,'F', 2200, 5),
('Manuel Matias',34563,'M', 2200,5);

-- povoamento Contacto_PS
INSERT INTO Contacto_PS (PS, Contacto) VALUES
(11234,'918273859'),
(98207,'928394017'),
(19981,'939492738'),
(56277,'969278123'),
(78352,'912346132'),
(34563,'967812349');

-- povoamento Utente
INSERT INTO Utente
(Nome,Nr,Sexo,DtNascimento,Profissao,Dador,FaixaEtaria,Rua,Cidade,Numero,DtObi
to) VALUES
('Rodolfinha Oliveira',137142,'F','1984/03/05','Professor',1,NULL,'Rua da
Porta','Caldas de Vizela',1,NULL),
('Sofia Ferreira',251938,'F','1976/09/06','Banqueiro',1,NULL,'Rua de Santa
Margarida','Braga',192,NULL),
('Nuno Dias',123456,'M','2007/02/05','Estudante',0,NULL,'Travessa das
Arroteias','Rio Tinto',180,NULL),

```

('Marco Silva',654678,'M','1992/08/22','Veterinário',1,NULL,'Rua Primeiro de Maio','Braga',245,NULL),
 ('Cristina Pinheiro',978675,'F','1943/11/12','Reformado',0,NULL,'Rua de Portugal','Valongo',58,NULL),
 ('Miquelina Maria',133564,'F','1947/01/15','Reformado',0,NULL,'Rua Gil Vicente','Guimarães',13,NULL),
 ('Francisco Manuel',995513,'M','2013/02/09','Estudante',0,NULL,'Rua Maximino de Matos','Fafe',18,NULL),
 ('José Joaquim',772413,'M','1968/05/28','Carpinteiro',1,NULL,'Rua Camilo Castelo Branco','Vila Nova de Famalicão',46,NULL),
 ('Guilhermina Sá Freitas',133897,'F','1929/10/17','Reformado',0,NULL,'Rua Santa Catarina','Porto',78,NULL),
 ('Manuel Antunes',353761,'M','1980/03/25','Escritorário',1,NULL,'Rua de Bom Jardim','Porto',33,NULL),
 ('Margarida Freitas',225839,'F','1988/04/27','Modista',1,NULL,'Rua da Misericórdia','Vila Verde',25,NULL),
 ('Pedro Frazão',667553,'M','1988/08/10','Madeireiro',1,NULL,'Rua da Igreja','Póvoa de Lanhoso',535,NULL),
 ('Aurora Menezes',180827,'F','1957/11/11','Professor',0,NULL,'Rua Padre Américo','Penafiel',16,NULL),
 ('Sónia Araújo',517943,'F','1992/09/25','Professor',1,NULL,'Praça da Repúblca','Caminha',10,NULL),
 ('Joana Pereira',693156,'F','2013/05/05','Estudante',0,NULL,'Rua das Barcas','Ponte Barca',57,NULL),
 ('António Moreira',954132,'M','1948/12/12','Reformado',0,NULL,'Rua São João de Deus','Vila Nova de Famalicão',31,NULL),
 ('Amélia Carmélia',576382,'F','1928/01/31','Reformado',0,NULL,'Rua São Marcos','Braga',140,NULL),
 ('Ameliana Carmo',576132,'F','1922/06/21','Reformado',0,NULL,'Avenida Central','Braga',40,'2018/11/18'),
 ('Joana Vaz',152637,'F','1960/10/19','Escritor',1,NULL,'Rua da Porta Nova','Braga',20,'2017/01/02'),
 ('Felisberto Castro',839271,'M','1940/03/05','Reformado',1,NULL,'Rua Cândido dos Reis','Barcelos',12,'2013/05/06');

```

-- povoamento Incapacidade
INSERT INTO Incapacidade (Id,Descricao) VALUES
(1, 'Esquizofrenia'), -- psiquiatra -- DADOR
(2, 'Toxicodependência'), -- NAO DADOR
(3, 'Etilismo Crónico'), -- psiquiatra -- NAO DADOR
(4, 'Depressão'), -- Psico -- DADOR PARA LIGEIRA
(5, 'Alterações Comportamentais'), -- Psico -- NAO DADOR
(6, 'Hiperatividade'), -- Psico, criança -- DADOR
(7, 'Obesidade'), -- N, MGF -- DADOR
(8, 'Intolerância Alimentar'), -- N -- DADOR
(9, 'Hipertensão'), -- MI, MFG -- DADOR
  
```

```
(10, 'Doença Pulmonar Obstrutiva Crónica'), -- MI -- NAO DADOR  
(11, 'Gripe'), -- mgf -- DADOR  
(12, 'Dores Abdominais'); -- mgf -- DADOR
```

```
-- povoamento Utente_Incapacitado  
INSERT INTO Utente_Incapacitado (Utente, Incapacidade) VALUES  
(667553,1),  
(137142,2),  
(180827,3),  
(517943,4),  
(123456,5),  
(995513,6),  
(137142,7),  
(693156,8),  
(133564,9),  
(954132,10),  
(251938,11),  
(995513,12);
```

```
-- povoamento Contacto_Utente  
INSERT INTO Contacto_Utente (Utente, Contacto) VALUES  
(137142,'919187762'),  
(251938,'965748219'),  
(123456,'928791236'),  
(654678,'938102941'),  
(978675,'255098789'),  
(978675,'918265910');
```

```
-- povoamento Consulta  
INSERT INTO Consulta (Id, Data,HoraEntrada,HoraSaida,Duracao,Descricao,Estado)  
VALUES -- ESTADO - 1: Marcada, 2: Iniciada, 3: Realizada, 4: Cancelada  
(1,'2018-11-01 12:15:00','2018-11-01 12:16:00',NULL,NULL,NULL,2),  
(2,'2022-01-03 08:15:00',NULL,NULL,NULL,NULL,1),  
(3,'2018-12-11 11:00:00','2018-12-11 11:05:00',NULL,NULL,NULL,2),  
(4,'2019-01-10 09:00:00',NULL,NULL,NULL,NULL,1),  
(5,'2018-04-05 18:30:00',NULL,NULL,NULL,NULL,4),  
(6,'2015-11-01          14:45:00','2015-11-01          14:46:00','2015-11-01  
15:01:00',NULL,'Utente com grau elevado de obesidade.Receita a prescrever.',3),  
(7,'2017-08-12          17:00:00','2017-08-12          17:00:00','2017-08-12  
17:10:00',NULL,'Utente com sintomas de intolerância alimentar.Receita a  
prescrever.',3),  
(8,'2018-04-22          10:15:00','2018-04-22          10:15:45','2018-04-22  
10:30:45',NULL,'Utente com hipertensão.Receita a prescrever.',3),
```

(9, '2018-09-15 16:45:00', '2018-09-15 16:46:00', '2018-09-15
 17:00:00', NULL, 'Utente apresenta sinais hiperatividade. Receita a prescrever.', 3),
 (10, '2021-02-02 11:45:00', NULL, NULL, NULL, NULL, 1),
 (11, '2021-03-15 12:00:00', NULL, NULL, NULL, NULL, 1),
 (12, '2017-06-25 19:45:00', '2017-06-25 19:45:30', '2017-06-25
 20:00:12', NULL, 'Utente com algumas alterações comportamentais. Receita a prescrever.', 3),
 (13, '2018-11-07 19:45:00', '2018-11-07 19:46:00', NULL, NULL, NULL, 2),
 (14, '2023-12-01 11:30:00', NULL, NULL, NULL, NULL, 1),
 (15, '2025-05-12 15:00:00', NULL, NULL, NULL, NULL, 4),
 (16, '2019-01-19 17:15:00', '2019-01-19 17:16:00', NULL, NULL, NULL, 1),
 (17, '2018-04-05 18:30:00', '2018-04-05 18:32:00', '2018-04-05
 18:49:00', NULL, 'Utente apresenta DPOC grave. Receita a prescrever. Necessidade de exames.', 3),
 (18, '2021-02-02 16:45:00', NULL, NULL, NULL, NULL, 1),
 (19, '2021-03-15 12:00:00', NULL, NULL, NULL, NULL, 4),
 (20, '2023-12-01 11:30:00', NULL, NULL, NULL, NULL, 1),
 (21, '2023-12-01 11:45:00', NULL, NULL, NULL, NULL, 1),
 (22, '2021-03-15 12:15:00', NULL, NULL, NULL, NULL, 4),
 (23, '2018-11-01 12:30:00', '2018-11-01 12:30:20', '2018-11-01
 12:45:50', NULL, 'Utente apresenta queixas de dores abdominais. Receita a prescrever.', 3),
 (24, '2018-10-11 08:30:00', '2018-10-11 08:30:00', '2018-10-11
 08:43:00', NULL, 'Utente com etilismo crónico. Receita a prescrever.', 3),
 (25, '2018-05-21 13:00:00', '2018-05-21 13:05:00', '2018-05-21
 13:25:00', NULL, 'Utente apresenta indícios de depressão. Receita a prescrever.', 3),
 (26, '2018-07-15 18:45:00', '2018-07-15 18:46:00', '2018-07-15
 18:55:10', NULL, 'Utente apresenta toxicodependência. Receita a prescrever.', 3),
 (27, '2018-01-11 10:45:00', '2018-01-11 10:47:00', '2018-01-11
 11:01:00', NULL, 'Utente encontra-se com gripe. Prescrição de receita.', 3),
 (28, '2018-04-13 19:00:00', '2018-04-13 19:01:00', '2018-04-13
 19:15:00', NULL, 'Utente com indícios de esquizofrenia. Necessidade de exames. Receita a prescrever.', 3),
 (29, '2018-09-13 14:15:00', '2018-09-13 14:15:00', '2018-09-13
 14:25:00', NULL, 'Consulta de Rotina', 3),
 (30, '2018-01-11 08:45:00', '2018-01-11 08:45:00', '2018-01-11
 09:00:00', NULL, 'Consulta de Rotina', 3),
 (31, '2018-03-12 09:15:00', '2018-03-12 09:15:00', '2018-03-12
 09:30:00', NULL, 'Consulta de Rotina', 3),
 (32, '2018-05-20 08:45:00', '2018-05-20 08:45:00', '2018-05-20
 09:00:00', NULL, 'Consulta de Rotina', 3);

-- povoamento Receita

```

INSERT INTO Receita (Id, Data, Descricao, Consulta) VALUES
(1, '2015-11-01', '3 vezes por dia', 6),
(2, '2017-08-12', '8 em 8 horas', 7),
(3, '2018-04-22', '8 em 8 horas', 8),
(4, '2018-09-15', '3 vezes por dia', 9),
(5, '2017-06-25', '1 vez por dia', 12),
(6, '2018-04-05', '3 vezes por dia, 15 minutos antes das refeições', 17),
(7, '2018-11-01', '1 vez por semana', 23),
(8, '2018-10-11', '1 vez por dia', 24),
(9, '2018-05-21', '1 vez por dia', 25),
(10, '2018-07-15', '1 vez por dia', 26),
(11, '2018-01-11', '3 vezes por dia', 27),
(12, '2018-04-13', '1 vez por dia', 27),
(13, '2018-01-11', '3 vezes por dia', 28);

```

-- povoamento Medicamento

```

INSERT INTO Medicamento(Id, Nome, Descricao) VALUES
(1, 'Haloperidol', '5mg comprimido'), -- esquizofrenia
(2, 'Butil-Escopolamina', 'supositório'), -- dores abdominais (criança)
(3, 'Paracetamol', '1000mg comprimido'),
(4, 'Brometo de Hipratrópio', 'nebulização'), -- doença pulmonar
(5, 'Losartan', '50mg comprimido'),
(6, 'Domperidona', '10mg comprimido'),
(7, 'Dulaglutida', '1.5mg Ampola subcutanea'),
(8, 'Metilfedinato', '60mg comprimido'),
(9, 'Nisperidona', '1mg comprimido'),
(10, 'Paroxetina', '20mg comprimido'),
(11, 'Tiapride', '100mg comprimido'),
(12, 'Metadona', '30mg comprimido'),
(13, 'Ibuprofeno', '400mg comprimido');

```

-- povoamento Medicamento_Receita

```

INSERT INTO Medicamento_Receita(Receita, Medicamento, Posologia) VALUES
(1, 1, 3),
(12, 2, 3),
(12, 3, 3),
(11, 3, 3),
(10, 4, 3),
(9, 5, 1),
(8, 6, 3),
(7, 7, 1),
(6, 8, 1),
(5, 9, 1),
(4, 10, 1),

```

```

(3,11,3),
(13,13,3),
(2,12,1);

-- povoamento PUC
INSERT INTO PUC(Utente,PS,Consulta,Preco) VALUES
(137142,78352,6,5),
(693156,98207,7,5),
(133564,34563,8,5),
(995513,56277,9,5),
(123456,56277,12,5),
(954132,11234,17,5),
(995513,78352,23,5),
(180827,19981,24,5),
(517943,56277,25,5),
(137142,19981,26,5),
(251938,34563,27,5),
(667553,19981,28,5),
(654678,34563,29,5),
(978675,78352,1,5),
(772413,11234,2,5),
(133897,11234,3,5),
(353761,98207,4,5),
(225839,98207,5,5),
(576382,34563,10,5),
(995513,56277,11,5),
(137142,78352,13,5),
(995513,78352,14,5),
(576382,34563,15,5),
(693156,98207,16,5),
(772413,11234,18,5),
(654678,34563,19,5),
(654678,34563,20,5),
(133897,11234,21,5),
(225839,19981,22,5),
(137142,78352,30,5),
(137142,78352,31,5),
(137142,78352,32,5);

```

Anexo III – Tabelas Povoadas

	Id	HoraSaida	Descricao	Duracao	Data	HoraEntrada	Estado
►	1	NULL	NULL	NULL	2018-11-01 12:15:00	2018-11-01 12:16:00	2
	2	NULL	NULL	NULL	2022-01-03 08:15:00	NULL	1
	3	NULL	NULL	NULL	2018-12-11 11:00:00	2018-12-11 11:05:00	2
	4	NULL	NULL	NULL	2019-01-10 09:00:00	NULL	1
	5	NULL	NULL	NULL	2018-04-05 18:30:00	NULL	4
	6	2015-11-01 15:01:00	Utente com grau elevado de obesidade. Receita...	NULL	2015-11-01 14:45:00	2015-11-01 14:46:00	3
	7	2017-08-12 17:10:00	Utente com sintomas de intolerância alimentar....	NULL	2017-08-12 17:00:00	2017-08-12 17:00:00	3
	8	2018-04-22 10:30:45	Utente com hipertensão. Receita a prescrever.	NULL	2018-04-22 10:15:00	2018-04-22 10:15:45	3
	9	2018-09-15 17:00:00	Utente apresenta sinais hiperatividade. Receita...	NULL	2018-09-15 16:45:00	2018-09-15 16:46:00	3
	10	NULL	NULL	NULL	2021-02-02 11:45:00	NULL	1
	11	NULL	NULL	NULL	2021-03-15 12:00:00	NULL	1
	12	2017-06-25 19:45:30	Utente com algumas alterações comportamentais...	NULL	2017-06-25 19:45:00	2017-06-25 19:45:30	3
	13	NULL	NULL	NULL	2018-11-07 19:45:00	2018-11-07 19:46:00	2
	14	NULL	NULL	NULL	2023-12-01 11:30:00	NULL	1
	15	NULL	NULL	NULL	2025-05-12 15:00:00	NULL	4
	16	NULL	NULL	NULL	2019-01-19 17:15:00	2019-01-19 17:16:00	1
	17	2018-04-05 18:49:00	Utente apresenta DPOC grave. Receita a presc...	NULL	2018-04-05 18:30:00	2018-04-05 18:32:00	3
	18	NULL	NULL	NULL	2021-02-02 16:45:00	NULL	1
	19	NULL	NULL	NULL	2021-03-15 12:00:00	NULL	4
	20	NULL	NULL	NULL	2023-12-01 11:30:00	NULL	1
	21	NULL	NULL	NULL	2023-12-01 11:45:00	NULL	1
	22	NULL	NULL	NULL	2021-03-15 12:15:00	NULL	4
	23	2018-11-01 12:45:50	Utente apresenta queixas de dores abdominais...	NULL	2018-11-01 12:30:00	2018-11-01 12:30:20	3
	24	2018-10-11 08:43:00	Utente com estílismo crónico. Receita a prescrever.	NULL	2018-10-11 08:30:00	2018-10-11 08:30:00	3
	25	2018-05-21 13:25:00	Utente apresenta indícios de depressão. Receita...	NULL	2018-05-21 13:00:00	2018-05-21 13:05:00	3
	26	2018-07-15 18:55:10	Utente apresenta toxicodependência. Receita a...	NULL	2018-07-15 18:45:00	2018-07-15 18:46:00	3
	27	2018-01-11 11:01:00	Utente encontra-se com gripe. Prescrição de re...	NULL	2018-01-11 10:45:00	2018-01-11 10:47:00	3
	28	2018-04-13 19:15:00	Utente com indícios de esquizofrenia. Necessida...	NULL	2018-04-13 19:00:00	2018-04-13 19:01:00	3
	29	2018-09-13 14:25:00	Consulta de Rotina	NULL	2018-09-13 14:15:00	2018-09-13 14:15:00	3
	30	2018-01-11 09:00:00	Consulta de Rotina	NULL	2018-01-11 08:45:00	2018-01-11 08:45:00	3
	31	2018-03-12 09:30:00	Consulta de Rotina	NULL	2018-03-12 09:15:00	2018-03-12 09:15:00	3
	32	2018-05-20 09:00:00	Consulta de Rotina	NULL	2018-05-20 08:45:00	2018-05-20 08:45:00	3

Figura 52 – Conteúdo tabela consulta

	PS	Contacto
►	11234	918273859
	19981	939492738
	34563	967812349
	56277	969278123
	78352	912346132
	98207	928394017

Figura 53 - Conteúdo tabela contacto_PS

	Utente	Contacto
►	123456	928791236
	137142	919187762
	251938	965748219
	654678	938102941
	978675	255098789
	978675	918265910

Figura 54 - Conteúdo tabela contacto_Utente

	Id	Descricao
►	1	Medicina Interna
	2	Nutrição
	3	Psiquiatria
	4	Psicologia
	5	Medicina Geral e Familiar

Figura 55 - Conteúdo tabela especialidade

	Id	Descricao
►	1	Esquizofrenia
	2	Toxicodependência
	3	Etilismo Crónico
	4	Depressão
	5	Alterações Comportamentais
	6	Hiperatividade
	7	Obesidade
	8	Intolerância Alimentar
	9	Hipertensão
	10	Doença Pulmonar Obstrutiva Crónica
	11	Gripe
	12	Dores Abdominais

Figura 56 - Conteúdo tabela incapacidade

	Medicamento	Receita	Posologia
►	1	1	3
	2	12	3
	3	11	3
	3	12	3
	4	10	3
	5	9	1
	6	8	3
	7	7	1
	8	6	1
	9	5	1
	10	4	1
	11	3	3
	12	2	1
	13	13	3

Figura 57 - Conteúdo tabela medicamento_receita

	Id	Nome	Descrição
▶	1	Haloperidol	5mg comprimido
	2	Butil-Escopolamina	suposório
	3	Paracetamol	1000mg comprimido
	4	Brometo de Hipratópio	nebulização
	5	Losartan	50mg comprimido
	6	Domperidona	10mg comprimido
	7	Dulaglutida	1.5mg Ampola subcutanea
	8	Metilfedinato	60mg comprimido
	9	Nisperidona	1mg compimido
	10	Paroxetina	20mg comprimido
	11	Tiapride	100mg comprimido
	12	Metadona	30mg comprimido
	13	Ibuprofeno	400mg comprimido

Figura 58 - Conteúdo tabela medicamento

	CC	Nome	Sexo	Salario	Especialidade
▶	11234	Roberto Cunha	M	2500	1
	19981	Diogo Magalhães	M	1400	3
	34563	Manuel Matias	M	2200	5
	56277	Maria Pinho	F	2900	4
	78352	Feliciana Freitas	F	2200	5
	98207	Anastácia Pereira	F	2000	2

Figura 59 - Conteúdo tabela profissional_saude

	Id	Data	Descrição	Consulta
▶	1	2015-11-01	3 vezes por dia	6
	2	2017-08-12	8 em 8 horas	7
	3	2018-04-22	8 em 8 horas	8
	4	2018-09-15	3 vezes por dia	9
	5	2017-06-25	1 vez por dia	12
	6	2018-04-05	3 vezes por dia, 15 minutos antes das refeições	17
	7	2018-11-01	1 vez por semana	23
	8	2018-10-11	1 vez por dia	24
	9	2018-05-21	1 vez por dia	25
	10	2018-07-15	1 vez por dia	26
	11	2018-01-11	3 vezes por dia	27
	12	2018-04-13	1 vez por dia	27
	13	2018-01-11	3 vezes por dia	28

Figura 60 - Conteúdo tabela receita

	Utente	PS	Consulta	Preco
►	123456	56277	12	5
	133564	34563	8	0
	133897	11234	3	0
	133897	11234	21	0
	137142	19981	26	4
	137142	78352	6	4
	137142	78352	13	4
	137142	78352	30	4
	137142	78352	31	4
	137142	78352	32	4
	180827	19981	24	5
	225839	19981	22	4
	225839	98207	5	4
	251938	34563	27	4
	353761	98207	4	4
	517943	56277	25	4
	576382	34563	10	0
	576382	34563	15	0
	654678	34563	19	4
	654678	34563	20	4
	654678	34563	29	4
	667553	19981	28	4
	693156	98207	7	5
	693156	98207	16	5
	772413	11234	2	4
	772413	11234	18	4
	954132	11234	17	0
	978675	78352	1	0
	995513	56277	9	5
	995513	56277	11	5
	995513	78352	14	5
	995513	78352	23	5

Figura 61 - Conteúdo tabela puc

	Incapacidade	Utente
►	5	123456
	9	133564
	2	137142
	7	137142
	3	180827
	11	251938
	4	517943
	1	667553
	8	693156
	10	954132
	6	995513
	12	995513

Figura 62 - Conteúdo tabela utente_incapacitado

Nr	Profissao	Nome	Sexo	Dador	DtNascimento	FaixaEtaria	Rua	Cidade	Numero	DtObito
123456	Estudante	Nuno Dias	M	0	2007-02-05	J	Travessa das Arroteias	Rio Tinto	180	NULL
133564	Reformado	Miquelina Maria	F	0	1947-01-15	S	Rua Gil Vincente	Guimarães	13	NULL
133897	Reformado	Gulhermina Sá Freitas	F	0	1929-10-17	S	Rua Santa Catarina	Porto	78	NULL
137142	Professor	Rodolfina Oliveira	F	1	1984-03-05	A	Rua da Porta	Caldas de Vizela	1	NULL
152637	Escritor	Joana Vaz	F	1	1960-10-19	A	Rua da Porta Nova	Braga	20	2017-01-02
180827	Professor	Aurora Menezes	F	0	1957-11-11	A	Rua Padre Américo	Penafiel	16	NULL
225839	Modista	Margarida Freitas	F	1	1988-04-27	A	Rua da Misericórdia	Vila Verde	25	NULL
251938	Banqueiro	Sofia Ferreira	F	1	1976-09-06	A	Rua de Santa Margarida	Braga	192	NULL
353761	Escrivário	Manuel Antunes	M	1	1980-03-25	A	Rua de Bom Jardim	Porto	33	NULL
517943	Professor	Sónia Araújo	F	1	1992-09-25	A	Praça da Repúblca	Caminha	10	NULL
576132	Reformado	Amélia Carmo	F	0	1922-06-21	S	Avenida Central	Braga	40	2018-11-18
576382	Reformado	Amélia Carmélia	F	0	1928-01-31	S	Rua São Marcos	Braga	140	NULL
654678	Veterinário	Marco Silva	M	1	1992-08-22	A	Rua Primeiro de Maio	Braga	245	NULL
667553	Madeireiro	Pedro Frazão	M	1	1988-08-10	A	Rua da Igreja	Póvoa de Lanh...	535	NULL
693156	Estudante	Joana Pereira	F	0	2013-05-05	J	Rua das Barcas	Ponte Barca	57	NULL
772413	Carpinteiro	José Joaquim	M	1	1968-05-28	A	Rua Camilo Castelo Bra...	Vila Nova de Fa...	46	NULL
839271	Reformado	Felisberto Castro	M	1	1940-03-05	S	Rua Cândido dos Reis	Barcelos	12	2013-05-06
954132	Reformado	António Moreira	M	0	1948-12-12	S	Rua São João de Deus	Vila Nova de Fa...	31	NULL
978675	Reformado	Cristina Pinheiro	F	0	1943-11-12	S	Rua de Portugal	Valongo	58	NULL
995513	Estudante	Francisco Manuel	M	0	2013-02-09	J	Rua Maximino de Matos	Fafe	18	NULL

Figura 63 - Conteúdo tabela utente

Anexo IV – Código Relativo a Interrogações e Transações

```
USE `centrosaude`;
```

```
-- Implementação de um trigger que, antes de ser inserido um utente, indica o valor da sua faixa etária
```

```
DELIMITER $$
```

```
CREATE TRIGGER AtualizaFaixaEtaria BEFORE INSERT ON Utente
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF (TIMESTAMPDIFF(YEAR, NEW.DtNascimento, CURDATE()) < 18) THEN
```

```
        SET NEW.FaixaEtaria = 'J';
```

```
    ELSE IF (TIMESTAMPDIFF(YEAR, NEW.DtNascimento, CURDATE()) > 65) THEN
```

```
        SET NEW.FaixaEtaria = 'S';
```

```
    ELSE SET NEW.FaixaEtaria = 'A';
```

```
END IF;
```

```
END IF;
```

```
END $$;
```

```
DELIMITER ;
```

```
-- Implementação de um procedure que insere um utente na BD
```

```
DELIMITER $$
```

```
CREATE PROCEDURE InserirUtente(NrUtente INT, Profissao VARCHAR(30), Nome VARCHAR(75), Sexo VARCHAR(1), Dador BOOLEAN, Data_Nascimento DATE, Rua VARCHAR(50), Cidade VARCHAR(50), Numero INT)
```

```
BEGIN
```

```
    DECLARE Erro BOOLEAN DEFAULT 0;
```

```
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;
```

```
    START TRANSACTION;
```

```

    INSERT INTO Utente (Nr, Profissao, Nome, Sexo, Dador, DtNascimento, Rua,
    Cidade, Numero) VALUES (NrUtente, Profissao, Nome, Sexo, Dador,
    Data_Nascimento, Rua, Cidade, Numero);

    IF erro = 1
    THEN
        BEGIN
        ROLLBACK;
        SELECT 'Sexo indicado pode não ser válido';
        END;
    ELSE COMMIT;
    END IF;

END $$

DELIMITER ;

-- Implementação de um procedure que insere um PS na BD
DELIMITER $$

CREATE PROCEDURE InserirPS(CC INT, Nome VARCHAR(100), Sexo VARCHAR(1), Salario
INT, Especialidade INT)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;
    INSERT INTO Profissional_Saude VALUES (CC, Nome, Sexo, Salario,
Especialidade);

    IF erro = 1
    THEN
        BEGIN
        ROLLBACK;
        SELECT 'Sexo indicado pode não ser válido';
        END;
    ELSE COMMIT;
    END IF;
END $$

DELIMITER ;

```

-- Implementação de um procedure que regista a morte de um utente

```

DELIMITER $$

CREATE PROCEDURE ObitoUtente(NrUtente INT, Data_Morte DATE)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

```

```

START TRANSACTION;

UPDATE Utente AS U SET U.dtObito=Data_Morte WHERE U.Nr=NrUtente;

IF erro = 1
THEN ROLLBACK;
ELSE COMMIT;
END IF;
END $$

DELIMITER ;

-- Implementação de um trigger que desmarca consultas de utente que morreu
-- ESTADO - 1: Marcada, 2: Iniciada, 3: Realizada, 4: Cancelada
DELIMITER $
CREATE TRIGGER DesmarcarObito BEFORE UPDATE ON Utente
FOR EACH ROW
DesmarcarObitoLabel:BEGIN
    IF new.DtObito IS NOT NULL THEN
        UPDATE Consulta
        INNER JOIN PUC ON PUC.Consulta = Consulta.Id
        SET Estado = 4
        WHERE Utente=old.Nr AND (Estado = 1 OR Estado
= 2);
    ELSE
        LEAVE DesmarcarObitoLabel;
    END IF ;
END $
DELIMITER ;

-- Implementação de um procedure para marcar consultas
DELIMITER $$

CREATE PROCEDURE MarcarConsulta(NrUtente INT, NrPS INT,NrConsulta INT, Data
DATETIME)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;

    IF ((SELECT DtObito FROM Utente WHERE Nr = NrUtente) IS NOT NULL) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Nao é possivel agendar
a consulta - Utente faleceu';
    ELSE
        IF ((SELECT SemSobreposicao(NrUtente, NrPS, Data)) > 0) THEN

```

```

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Nao é possivel agendar
a consulta - Sobreposição de consultas';
    ELSE
        INSERT INTO Consulta (Id, Data, Estado) VALUES (NrConsulta,Data,1);
        INSERT INTO PUC (Utente, PS, Consulta)VALUES (NrUtente, NrPS,
NrConsulta);
    END IF;
END IF;

IF erro = 1
THEN BEGIN
    ROLLBACK;
    SELECT 'Possibilidades: Utente faleceu; existe sobreposição de
consultas; horário de agendamento não é possível';
    END;
ELSE COMMIT;
END IF;
END $$

DELIMITER ;

-- Implementação de um procedimento para desmarcar consultas
DELIMITER $$

CREATE PROCEDURE DesmarcarConsulta(NrUtente INT, data DATETIME)
BEGIN

DECLARE Erro BOOLEAN DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

START TRANSACTION;
UPDATE Consulta
    INNER JOIN PUC ON PUC.Consulta = Consulta.Id
    SET Estado = 4
    WHERE Utente=NrUtente AND Data=data;
IF erro = 1
THEN
    BEGIN
    ROLLBACK;
    SELECT 'Possibilidades: Consulta já está desmarcada';
    END;
ELSE COMMIT;
END IF;
END $$

DELIMITER ;

```

```

-- Implementação de uma função para ajudar a agendar uma consulta sem
sobreposição
DELIMITER $$

CREATE FUNCTION SemSobreposicao (ut INT, ps INT, dt DATETIME) RETURNS INT
DETERMINISTIC
BEGIN
    RETURN (SELECT COUNT(*) FROM Consulta INNER JOIN PUC
ON (PUC.Consulta = Consulta.Id)
WHERE Data
BETWEEN SUBTIME(dt, TIME('00:14:59'))
AND
ADDTIME(dt, TIME('00:14:59')))

    AND Consulta.Estado <> 3 AND Consulta.Estado <> 4

    AND (PUC.Utente = ut OR PUC.PS = ps OR (PUC.Utente = ut AND
PUC.PS = ps));
END $$

DELIMITER ;


-- Implementação de um procedure que permite remarcar uma consulta : Cancela a
antiga e cria uma nova
DELIMITER $$

CREATE PROCEDURE RemarcarConsulta(NrUtente INT, data_velha DATETIME, data_nova
DATETIME, NrPS INT, NrConsulta INT)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;

    CALL DesmarcarConsulta(NrUtente, data_velha);
    CALL MarcarConsulta(NrUtente, NrPS, NrConsulta, data_nova);

    IF erro = 1
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END $$

DELIMITER ;


-- Implementação de um trigger que permite calcular o preço da consulta
consoante os parâmetros: Faixa Etária e Dador
DELIMITER $

CREATE TRIGGER precoConsulta BEFORE INSERT ON PUC

```

```

FOR EACH ROW
BEGIN
    IF ((SELECT FaixaEtaria FROM Utente WHERE Nr = new.Utente) = 'S') THEN SET
new.Preco = 0;
        ELSE IF ((SELECT Dador FROM Utente WHERE Nr = new.Utente) = 1)
THEN SET new.Preco = 4;
            ELSE SET new.Preco = 5;
        END IF;
    END IF;
END $$
DELIMITER ;

-- Implementação de um trigger que calcula a duração da consulta em Update
DELIMITER $$
CREATE TRIGGER MeterDuracaoU BEFORE UPDATE ON Consulta
FOR EACH ROW
BEGIN
    IF new.HoraSaida IS NOT NULL THEN
        SET new.Duracao =
TIMESTAMPDIFF(MINUTE,new.HoraEntrada,new.HoraSaida);
    END IF;
END $$$
DELIMITER ;
-- Implementação de um trigger que calcula a duração da consulta em Insert
DELIMITER $$
CREATE TRIGGER MeterDuracaoI BEFORE INSERT ON Consulta
FOR EACH ROW
BEGIN
    IF new.HoraSaida IS NOT NULL THEN
        SET new.Duracao =
TIMESTAMPDIFF(MINUTE,new.HoraEntrada,new.HoraSaida);
    END IF;
END $$$
DELIMITER ;

-- Implementação de um trigger que atualiza o estado
DELIMITER $$
CREATE TRIGGER AtualizarEstado BEFORE UPDATE ON Consulta
FOR EACH ROW
BEGIN
    IF new.HoraEntrada IS NOT NULL THEN
        IF new.Estado = 1 THEN
            SET new.Estado = 2;
        ELSE IF new.Estado = 2 AND new.HoraSaida IS NOT NULL THEN
            SET new.Estado = 3;
        END IF;
    END IF;
END $$$
DELIMITER ;

```

```

        END IF;
    END IF;
END $$

DELIMITER ;

-- Implementação de um procedure que devolve a lista de receitas emitidas por
determinado PS
DELIMITER $
CREATE PROCEDURE ReceitasM(idM INT)
BEGIN
    SELECT R.Id,R.Descricao
        FROM (( Receita R
                INNER JOIN Consulta C ON R.Consulta = C.Id)
                INNER JOIN PUC P ON P.Consulta = C.id)
        WHERE P.PS = idM;
END $$

DELIMITER ;

-- Implementação de um procedure que calcula o numero de receitas emitidas por
determinado PS
DELIMITER $
CREATE PROCEDURE NmReceitasM(idM INT)
BEGIN
    SELECT COUNT(R.Id)
        FROM (( Receita R
                INNER JOIN Consulta C ON R.Consulta = C.Id)
                INNER JOIN PUC P ON P.Consulta = C.id)
        WHERE P.PS = idM;
END $$

DELIMITER ;

-- Implementação de um procedure que calcula quanto o CS ganhou num mês
DELIMITER $
CREATE PROCEDURE DinheiroMes(mes INT,ano INT)
BEGIN
    SELECT SUM(P.Preco)
        FROM ( PUC P
                INNER JOIN Consulta C ON P.Consulta = C.id)
        WHERE ((YEAR(C.Data) = ano) AND (MONTH(C.Data) = mes) AND
(C.Estado = 3));
END $$

DELIMITER ;

-- Implementação de um procedure que calcula o numero de consultas feitas em
determinado tempo
DELIMITER $

```

```

CREATE PROCEDURE ConsultaTemp(inicio DATETIME,fim DATETIME)
BEGIN
    SELECT COUNT(C.Id)
    FROM Consulta C
        WHERE ((C.Data > inicio) AND (C.Data < fim));
END $$

DELIMITER ;

-- Implementação de um procedure que devolve as receitas que foram emitidas
-- numa consulta de um dado utente
DELIMITER $
CREATE PROCEDURE ReceitasEmConsulta(idU INT,dataC DATETIME)
BEGIN
    SELECT R.Id,R.Descricao
    FROM ((Receita R
            INNER JOIN Consulta C ON C.Id = R.Consulta)
            INNER JOIN PUC P ON P.Consulta = C.Id)
            WHERE ((P.Utente = idU) AND (C.Data = dataC));
END $$

DELIMITER ;

-- Implementação um procedure que, dado o ID de um profissional de saúde,
-- apresente as consultas feitas
DELIMITER $$

CREATE PROCEDURE recebeConsulta (IN numero INT)
READS SQL DATA
BEGIN
    SELECT PC.Consulta AS NumConsulta, P.CC AS NumCC
    FROM ((PUC PC
            INNER JOIN Profissional_Saude P ON P.CC = PC.PS)
            INNER JOIN Consulta C ON C.Id = PC.Consulta)
            WHERE P.CC = numero
            ORDER BY PC.Consulta;
END $$

DELIMITER $$

-- Implementação um procedure que, dado um ID de um profissional de saúde,
-- apresente as consultas que realizou em determinado dia
DELIMITER $$

CREATE PROCEDURE consultaDiaria (IN numero INT, dia DATE)
READS SQL DATA
BEGIN
    SELECT PC.Consulta AS Numero, P.CC AS Numero, C.Data AS Dia
    FROM ((PUC PC
            INNER JOIN Profissional_Saude P ON P.CC = PC.PS)
            INNER JOIN Consulta C ON C.Id = PC.Consulta)

```

```

        WHERE P.CC = numero AND DATE(C.Data) = dia
        ORDER BY PC.Consulta;

END $$

DELIMITER $$

-- Implementação de uma function que indica qual o bónus anual que um PS vai
ganhar
DELIMITER $$

CREATE FUNCTION bonusS (Prof_S INT, ano YEAR) RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE bonus INT;
    DECLARE EXIT HANDLER FOR NOT FOUND RETURN NULL;
    SET bonus = (SELECT COUNT(*) FROM (SELECT
COUNT(PC.Consulta) AS Numero
        FROM (PUC PC
        INNER JOIN Consulta C ON C.Id =
PC.Consulta)
        WHERE YEAR(C.Data) = ano
        AND PC.PS = Prof_S
        GROUP BY
PC.Utente) AS T
        WHERE T.Numero >= 4);
    RETURN bonus*50;
END $$

DELIMITER ;

-- Implementação de um procedure que calcula o bónus de um PS ao final de um
ano
DELIMITER $$

CREATE PROCEDURE bonus (IN Prof_S INT, ano YEAR)
READS SQL DATA
BEGIN
    SELECT bonusS(Prof_S, ano) AS Bonus;
END $$

DELIMITER ;

-- Implementação um procedimento que, dado um ID de um utente, apresente o seu
histórico clínico
DELIMITER $$

CREATE PROCEDURE historicoClinico (IN numero INT)
READS SQL DATA
BEGIN
    SELECT U.Nr AS Numero, C.Descricao AS Descricao, C.Data AS Data

```

```

        FROM ((PUC PC
                INNER JOIN Utente U ON U.Nr = PC.Utente)
                INNER JOIN Consulta C ON C.Id = PC.Consulta)
        WHERE U.Nr = numero AND C.Estado = 3
        ORDER BY C.Data;

END $$

DELIMITER ;

-- Implementação uma view que apresente todas as pessoas que já faleceram
DROP VIEW IF EXISTS listaObito;
CREATE VIEW listaObito AS
SELECT U.Nome, U.Nr, U.DtObito
        FROM Utente U
        WHERE U.DtObito IS NOT NULL
        ORDER BY U.Nome ASC;

-- View que mostra info de todos os utentes
DROP VIEW IF EXISTS vwUtentes;
CREATE VIEW vwUtentes AS
SELECT DISTINCT Nr AS 'Identificação',
                Nome AS 'Nome',
                FaixaEtaria AS 'Faixa Etária',
                Dador AS 'Dador',
                Sexo AS 'Sexo'
                From Utente
        WHERE DtObito IS NULL;

-- View que mostra info de todos os medicamentos
DROP VIEW IF EXISTS vwMedicamentos;
CREATE VIEW vwMedicamentos AS
SELECT DISTINCT Id AS 'Identificação',
                Nome AS 'Nome',
                Descricao AS 'Informação Extra'
                FROM Medicamento;

-- Implementação um trigger que não permita que sejam marcadas consultas
durante as horas de encerramento do centro de saúde ou em dias festivos (Natal
e Ano Novo)
DELIMITER $$

CREATE TRIGGER AgendarEmTempoUtil BEFORE INSERT ON Consulta
FOR EACH ROW
BEGIN
    IF (TIME(new.Data) >= '20:00:00' OR TIME(new.Data) <= '07:59:59' )
        OR (MONTH(new.Data) = 12 AND DAY(new.Data) = 25 )

```

```

        OR (MONTH(new.Data) = 1 AND DAY(new.Data) = 1)
    THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Nao é possivel agendar a
consulta';
END IF;
END $$

DELIMITER ;

-- Implementação de um trigger que garante que não é inserido nenhum sexo
-- aquando o registo do Utente que não seja M ou F
DELIMITER $$

CREATE TRIGGER GenderSafe BEFORE INSERT ON Utente
FOR EACH ROW
BEGIN
    IF (new.Sexo <> 'F' AND new.Sexo <> 'M')
    THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Sexo escolhido não é
válido, use F ou M';
END IF;
END $$

DELIMITER ;

-- Implementação de um trigger que garante que não é inserido nenhum sexo
-- aquando o registo do PS que não seja M ou F
DELIMITER $$

CREATE TRIGGER GenderSafePS BEFORE INSERT ON Profissional_Saude
FOR EACH ROW
BEGIN
    IF (new.Sexo <> 'F' AND new.Sexo <> 'M')
    THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Sexo escolhido não é
válido, use F ou M';
END IF;
END $$

DELIMITER ;

-- Implementação um procedure que emita receitas
DELIMITER $$

CREATE PROCEDURE EmitirReceita(receita INT, data DATE, descricao VARCHAR(100),
consulta INT)
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;
    IF ((SELECT Estado FROM Consulta WHERE Id = consulta) <> 4) AND
((SELECT Estado FROM Consulta WHERE Id = consulta) <> 1)
    THEN INSERT INTO Receita (Id, Data, Descricao, Consulta)
VALUES (receita, data, descricao, consulta);

```

```

        ELSE
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Consulta está
desmarcada ou ainda não começou';
        END IF;
    IF erro = 1
    THEN ROLLBACK;
        BEGIN
        ROLLBACK;
        SELECT 'Consulta está desmarcada ou ainda não começou';
        END;
    ELSE COMMIT;
    END IF;
END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE ContactoUtente(Id INT, contacto VARCHAR(50))
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;
        INSERT INTO Contacto_Utente VALUES (Id, contacto);
    IF erro = 1
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE ContactoPs(CC INT, contacto VARCHAR(50))
BEGIN
    DECLARE Erro BOOLEAN DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;
        INSERT INTO Contacto_PS VALUES (Id, contacto);
    IF erro = 1
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END $$

DELIMITER ;

```

Anexo V – Script de migração dos dados de MySQL para MongoDB

```
import datetime
import mysql.connector
from pymongo import MongoClient

client      = MongoClient()
db          = client.test
collection = db.utentes

cnx         = mysql.connector.connect(
                           user      ='root',
                           password = '',
                           database ='CentroSaude')
cursor     = cnx.cursor()

# Utentes

row_titles = (
    '_Nr', 'Profissao', 'Nome',
    'Sexo', 'Dador', 'DtNascimento',
    'FaixaEtaria', 'Rua', 'Cidade', 'Numero',
    'DtObito', 'Incapacidades', 'Contactos'
)

query = ( "SELECT "
          "* FROM Utente" )

cursor.execute( query )

for ( row ) in cursor:
    collection.insert({"_id" : int(row[0]), "Profissao" : row[1], "Nome" : row[2], "Sexo" : row[3], "Dador" : int(row[4]), "DtNascimento" : str(row[5]), "FaixaEtaria" : row[6], "Rua" : row[7], "Cidade" : row[8], "Numero" : int(row[9]), "DtObito" : str(row[10])})

query2 = ("SELECT "
          "U.Nr, UC.Contacto "
          "FROM Utente U "
          "Inner Join Contacto_Utente UC "
          "ON UC.Utente = U.Nr"
          )
```

```

cursor.execute(query2)

for( row ) in cursor:
    collection.update({"_id" : int(row[0])}, {"$addToSet' : {"Contactos" : str(row[1])}})

query3 = ("SELECT "
          "U.Nr, I.Descricao "
          "FROM Utente U "
          "Inner Join Utente_Incapacitado UI "
          "ON U.Nr = UI.Utente "
          "Inner Join Incapacidade I "
          "On I.Id = UI.Incapacidade"
          )

cursor.execute(query3)

for( row ) in cursor:
    collection.update({"_id" : int(row[0])}, {"$addToSet' : {"Incapacidades" : str(row[1])}})

# Medicos
collection2 = db.profissionais

row_titles = (
    '_id', 'Nome', 'Sexo', 'Salario', 'Especialidade'
)

query4 = ( "SELECT "
          "P.CC, P.Nome, P.Sexo, P.Salario, E.Descricao "
          "FROM Profissional_Saude P "
          "INNER JOIN Especialidade E "
          "ON P.Especialidade = E.Id"
          )

cursor.execute( query4 )

for ( row ) in cursor:
    collection2.insert({"_id" : int(row[0]), "Nome" : row[1], "Sexo" : row[2],
    "Salario" : int(row[3]), "Especialidade" : str(row[4])})

```

```

query5 = ("SELECT "
    "P.CC, PC.Contacto "
    "FROM Profissional_Saudade P "
    "Inner Join ContacTo_PS PC "
    "ON PC.PS = P.CC"
    )

cursor.execute(query5)

for( row ) in cursor:
    collection2.update({"_id" : int(row[0])}, {"'$addToSet' : {"Contactos" : row[1]}})

# Consultas
collection3 = db.consultas

row_titles = (
    'Data', 'HoraEntrada',
    'HoraSaida', 'Duracao', 'Descricao', 'Estado', 'Preco',
    'Utente_Nr', 'Medico_CC'
)

query6 = ( "SELECT "
    "C.Id, C.Data, C.HoraEntrada, C.HoraSaida, C.Duracao, C.Descricao,
    C.Estado, P.Preco, P.Utente, P.PS "
    "FROM Consulta C INNER JOIN PUC P "
    "ON P.Consulta = C.Id"
    )

cursor.execute( query6 )

for ( row ) in cursor:

    collection3.insert({"_id" : int(row[0]), "Data" : str(row[1]), "HoraEntrada" :
    : str(row[2]), "HoraSaida" : str(row[3]), "Duracao" : str(row[4]), "Descricao" :
    : str(row[5]), "Estado" : int(row[6]), "Preco" : float(row[7]), "Utente_id" :
    : int(row[8]), "Medico_id" : int(row[9])})

# Receitas

query7 = ( "SELECT "

```

```

"C.Id AS Consulta, R.Id, M.Nome, M.Descricao, RC.Posologia "
"FROM Receita R "
    "Inner join Medicamento_Receita RC "
    "ON RC.Receita = R.Id "
        "Inner join Medicamento M "
        "ON M.Id = RC.Medicamento "
            "Inner join Consulta C "
            "ON R.Consulta = C.Id"
)

cursor.execute( query7 )

cus = dict()

for( row ) in cursor:
    collection3.update({"_id": row[0]}, {"$push": {"Receitas": {"Receita": (row[1]), "medicamento": (row[2]), "descricao": str(row[3]), "posologia": int(row[4])}}})

cursor.close()
cnx.close()

```