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

Unidade Curricular de Bases de Dados

Ano Letivo de 2019/2020

Clínica+

André Loureiro Morais – A83899

José Pedro Oliveira Silva – A84577

Luís Mário Macedo Ribeiro – A85954

Pedro Miguel Borges Rodrigues – A84783

Dezembro, 2019

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Clínica+

André Loureiro Morais – A83899

José Pedro Oliveira Silva – A84577

Luís Mário Macedo Ribeiro – A85954

Pedro Miguel Borges Rodrigues – A84783

Dezembro, 2019

Resumo

O presente trabalho foi realizado no âmbito da unidade curricular de Base de Dados e o mesmo consistiu na criação de uma base de dados relacional, tendo sido feita posteriormente a migração para uma base de dados não relacional. Em ambas o objetivo era demonstrar o funcionamento de uma clínica de testes médicos à qual decidimos dar o nome de **Clinica+**.

Numa fase inicial foi retratada a contextualização deste tema, qual a motivação e objetivos deste projeto, bem com a análise de viabilidade. Primeiramente, começamos por identificar as necessidades de uma clínica, efetuando assim o levantamento de requisitos. Após termos completado o levantamento de requisitos, realizamos a análise dos mesmos de modo a garantir que todos os pontos que tínhamos salientado estavam de acordo com o que era pretendido.

Posteriormente elaboramos o modelo conceptual. Neste são apresentadas todas as entidades relevantes ao projeto, assim como os seus atributos e a forma como estes se relacionam. Tendo em atenção os atributos constituintes de cada entidade, identificamos assim, as chaves candidatas. Após a validação deste modelo estamos agora prontos para começar o modelo lógico.

O modelo lógico tem como base o modelo conceptual, estando este assim na sua base de criação. Neste modelo conseguimos fazer uma abordagem mais profunda sobre as existentes entidades e os seus relacionamentos, assim como do domínio dos seus atributos de modo a identificar quais as chaves primárias e estrangeiras. Tal como nos outros modelos, também foi feita uma validação do modelo lógico, onde garantimos a integridade dos dados.

A última fase da elaboração da base de dados não relacional teve como objetivo a construção do modelo físico. Para tal, começamos por realizar a tradução do modelo elaborado previamente para um **SGBS**, implementando primeiramente um script de criação e povoamento das tabelas em SQL. Encontramo-nos assim, neste momento da realização do trabalho, com uma base de dados perfeitamente funcional, no entanto ficam a faltar maneiras de interagir com a mesma. Para isso, foram realizadas queries, tendo estas como objetivo satisfazer os pedidos provenientes de entidades que tenham contacto com a base de dados. Por fim, foram definidas as views e as suas regras de utilização.

Na segunda fase, começamos por exportar os dados elaborando um script em Java no qual extraímos os dados de cada tabela para um ficheiro json. Para a migração da BD implementada anteriormente para uma NoSQL recorremos à ferramenta mongoimport.

Terminamos com a apresentação das conclusões do trabalho realizado, elaborando uma análise crítica de todo o processo realizado anteriormente bem como sugestões para trabalho futuro.

Área de Aplicação: Desenho e arquitectura de Sistemas de Bases de Dados no âmbito de uma aplicação desenvolvida para registar os testes médicos realizados por atletas.

Palavras-Chave: Bases de Dados Relacional, Bases de Dados não Relacional, Análise de Requisitos, Modelo Conceptual, Modelo Lógico, Modelo Físico, MySQL Workbench, SQL, NoSQL, MongoDB, Migração, JAVA, JSON.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

André, Luís, José e Pedro são um grupo de amigos que praticaram atletismo no FCP desde os seis anos de idade. Quando eram mais novos, todos estes rapazes olhavam para o atletismo como uma atividade extracurricular com o objetivo de se divertirem, podendo lá estar com os amigos e ao mesmo tempo praticar exercício físico. No entanto à medida que os anos foram passando os objetivos ficaram diferentes, e já nenhum dos quatro amigos encarava o atletismo como apenas uma diversão, havia dentro deles uma vontade de vencer, vontade essa que os levava a serem melhores e a quererem sempre obter melhores resultados. Com o passar dos anos, os objetivos do grupo de amigos não foi a única coisa que evolui. À medida que estes iam subindo de escalão, o rigor nos testes clínicos que o clube exigia também acrescia.

O grupo de amigos apoiava esta medida pois concordavam que a saúde era uma prioridade e que todos os atletas deviam estar no mesmo nível de preparação, isto é, caso um atleta não estivesse apto para competir, devido ao resultado dos seus testes, este não deveria receber aprovação médica. Esta era a lógica pela qual eles se regiam, no entanto, na clínica na qual realizavam os testes, certos funcionários tinham uma perspetiva diferente acerca do assunto, onde o dinheiro falava mais alto do que o profissionalismo.

Inconformados com este tipo de atitudes, o grupo de amigos tomou uma decisão conjunta de contruir uma clínica de testes clínicos onde o profissionalismo está acima de tudo. Estavam todos decididos e por isso criaram a **"Clínica+"**.

Alguns meses passaram e o que era apenas um projeto em papel estava prestes a tornar-se realidade, a **Clínica+** foi inaugurada. Estava pela frente agora a tarefa mais difícil que o grupo teria de enfrentar, adquirir clientes para manter a clínica de portas abertas ao público.

Após debaterem por inúmeras horas o grupo chegou à brilhante ideia de recorrerem ao seu passado para adquirir clientes. Para isso, entraram em contacto com os seus companheiros de equipa, para que eles realizassem os testes na **Clínica+**. Assim foi, os colegas de equipa realizaram os testes e, devido ao feedback positivo que deixaram, os quatro amigos receberam a notícia de que o FCP estaria interessado em fazer uma parceria. Como seria expectável, tal oferta foi aceite e com isso veio o reconhecimento público que o grupo tanto queria, expandindo assim o seu negócio.

Tudo corria bem para os quatro amigos, no entanto, o que outrora era uma pequena clínica que tinha poucos clientes, estava agora a ter de lidar com centenas de clientes e por isso, o que antes funcionava com papel e caneta para agendar consultas era agora impensável. Com isto, o grupo decidiu recorrer aos nossos serviços para a elaboração de um Sistema de Bases de Dados capaz de suportar as consultas dos atletas.

1.2. Apresentação do Caso de Estudo

A **Clínica+** é uma empresa na área da realização de testes médicos que procura o melhor para os seus utentes, tratando estes com o maior dos profissionalismos. De modo a assegurar tal coisa, foram criados vários tipos de testes.

Esta clínica destaca-se por ter um ambiente acolhedor e ao mesmo tempo profissional. O ambiente profissional deve-se ao facto de que para cada tipo de teste existem médicos formados nessa especialidade e também devido ao facto de cada médico trabalhar por turnos, o que permite assegurar uma maior disponibilidade dos médicos face ao número de testes.

De modo a identificar o atleta, a clínica faz a recolha de alguns dados quando o mesmo tenta agendar os testes, sendo estes dados, o número do cartão de cidadão, o nome, a data de nascimento, o clube ao qual pertence, a sua nacionalidade, a sua morada, a modalidade e categoria, e por fim o seu contacto. Quando são revelados os resultados dos testes é determinado se o atleta em questão está apto ou não.

Tendo em conta os aspetos acima descritos, a clínica considera que o seu modelo de negócio é viável e pretende avançar com o desenvolvimento do mesmo, implementando uma base de dados que servirá de suporte ao sistema.

1.3. Motivação e Objetivos

Dado ao crescimento repentino da **Clínica+**, e ao facto de a mesma ainda estar em desenvolvimento, esta ainda não possui um sistema digital de armazenamento de dados. No entanto, devido ao seu crescimento, o grupo de amigos decidiu a necessidade de reverter esta situação e para tal recorreu aos nossos serviços para a implementação de uma base de dados.

Vendo nesta clínica uma boa oportunidade para demonstrar os nossos conhecimentos científicos, rapidamente aceitamos a proposta.

Esta base de dados assenta nos seguintes pressupostos:

- Maior facilidade em saber os testes clínicos que um atleta realizou, assim como o médico que o executou.
- Maior facilidade em manter o histórico de testes, podendo este ser individual ou por clube.
- Obter o número de profissionais que trabalham na clínica e na área em que são especialistas, de modo a poder fazer uma melhor gestão dos funcionários.

- Conhecer todos os tipos de testes que um atleta tem de fazer e obter um resultado positivo para estar apto.

1.4. Estrutura do Relatório

De modo a estruturar o relatório, decidimos fazer a divisão em 5 fases:

- Introdução
- Levantamento e análise de requisitos
- Implementação da base de dados relacional
    - Modelação conceptual
    - Modelação lógica
    - Implementação física
- Implementação da base de dados não relacional
- Conclusões

Como previamente apresentado, na fase inicial, existiu a contextualização do problema assim como a apresentação dos objetivos e motivação para a realização do projeto.

Na segunda fase, é realizada uma reunião com os representantes da administração da clínica com o objetivo a saber quais as necessidades da sua aplicação, resultando isto, num levantamento de requisitos aos quais a base de dados implementada terá de responder e satisfazer.

Por sua vez, na terceira fase, é demonstrado todo o processo a realizar para a implementação da nossa base de dados relacional. Aqui, é apresentada toda a metodologia, desde a modelação conceptual até à modelação lógica e física. Na conceção da modelação conceptual, são identificadas as entidades, os relacionamentos entre estas, os atributos das entidades e dos relacionamentos. É ainda realizada a validação deste modelo com o utilizador. Após isso, é apresentado o modelo lógico, sendo este composto pela derivação de relações do modelo lógico e validação destas através da normalização e ainda verificação das restrições de integridade. Por fim, é descrito o modelo físico onde são satisfeitos os requisitos através de queries interativas com sistema SQL.

Na quarta fase, é apresentado todo o processo da elaboração da base de dados não relacional, tendo esta sido implementada com base na base de dados relacional previamente descrita. Assim, mostramos como realizamos a migração dos dados para o MongoDB. É também demonstrada, a realização das queries definidas previamente.

Para terminar o nosso relatório, são apresentadas uma série de conclusões, bem como, uma análise crítica do que fora realizado e uma avaliação do projeto.

2. Levantamento e análise de requisitos

2.1. Método de levantamento e de análise de requisitos

De modo a obter a informação necessária e recolher os objetivos pretendidos no desenvolvimento da base de dados, foi realizada uma reunião com a administração da **Clínica+**.

2.2. Especificação de Requisitos

Com base na reunião realizada, identificaram-se os seguintes requisitos.

2.2.1 Requisitos de descrição

1. Os atletas devem ser registados sendo-lhes atribuído um ID sequencial pela aplicação (sendo este único) e deve ser fornecido nome, data de nascimento, clube, nacionalidade, morada, contacto e por fim a modalidade e categoria em que o mesmo compete.
2. Os médicos são registados atribuindo-lhes um ID, novamente, sequencial e devem ser fornecidos o nome e a especialidade da sua formação.
3. Os testes são registados através da atribuição de um ID sequencial sendo necessário fornecer a data para qual estão agendados, o identificador do médico que executará o teste e o identificador do atleta que realizará o teste.
4. Cada turno é identificado por um ID sequencial que lhe é atribuído, sendo apenas necessário fornecer a designação, a hora de início e a hora de fim para fazer o seu registo.
5. Tanto a modalidade como a categoria que um atleta frequenta são identificadas por um ID e é necessário fornecer-lhes uma designação.
6. Uma especialidade é identificada pelo seu ID sequencial sendo necessário atribui-lhe uma designação e um preço para fazer o registo da mesma.

7. Um atleta pode realizar múltiplos testes e apenas quando passar em todos se encontra apto para competir.
8. Um teste é realizado por apenas um médico.
9. Cada médico tem um turno de trabalho
10. Cada médico tem apenas uma especialidade.
11. Uma modalidade pode ter várias categorias.
12. Cada atleta pode competir apenas numa modalidade.

2.2.2 Requisitos de exploração

Do ponto de vista do utilizador/atleta deve ser possível:

1. Verificar as próximas consultas
2. Ver os médicos disponíveis por turno
3. Ver os resultados de um teste
4. Consultar histórico de consultas
5. Visualizar todos os testes realizados por especialidade

Do ponto de vista do administrador:

6. Obter o número total de atletas
7. Obter todos os médicos que trabalham na clínica
8. Obter número de médicos por especialidade
9. Consultar o total faturado desde a abertura da clínica
10. Obter o total faturado por especialidade
11. Obter o total faturado por mês
12. Verificar a percentagem de aprovação por especialidade
13. Verificar a percentagem de aprovação por modalidade e categoria
14. Obter o número de testes executados por médico
15. Obter o número de testes executados por especialidade
16. Obter por cada turno o número de médicos
17. Obter por cada turno o número de testes realizados
18. Obter os resultados de todos os testes realizados
19. Consultar os próximos testes de um médico
20. Consultar os próximos testes de um atleta
21. Verificar os resultados dos atletas de um clube
22. Verificar os próximos testes dos atletas de um clube
23. Apresentar médicos e turnos

2.2.3 Requisitos de controlo

Do ponto de vista de um utilizador/atleta, este deve conseguir:

- Inserir os seus dados de registo
- Atualizar os seus dados de registo
- Permissões de consulta das informações relativas aos seus testes

Do ponto de vista de um clube, este deve conseguir:

- Inserir os dados dos atletas pertencentes ao clube
- Atualizar os dados dos atletas pertencentes ao clube
- Inserir atleta
- Atualizar categoria dos atletas pertencentes ao clube
- Permissões de consulta das informações relativas aos testes dos seus atletas

Do ponto de vista de um funcionário, este deve conseguir:

- Adiciona testes a um determinado atleta
- Atualiza/Cancela um teste anteriormente adicionado
- Atualiza data de um teste
- Atualiza especialidade de um teste
- Permissões de consulta das informações relativas a todos os testes e atletas

Do ponto de vista de um médico, este deve conseguir:

- Executa processos de concluir e cancelar testes
- Permissões de consulta das informações relativas a todos os testes e dados de atletas

Do ponto de vista de um administrador, este deve conseguir:

- Os administradores da Clínica+ tem permissões de consulta, inserção e alteração em todas as entidades restantes.

2.3. Análise dos requisitos

Após a realização dos requisitos, reunimos novamente com a administração da Clínica+ com o objetivo de ter a validação sobre os mesmos. Após analisarem os requisitos apresentados, a administração validou os mesmos, dando assim condições para prosseguirmos com o nosso trabalho. Assim sendo, começamos a construção do modelo conceptual suportado pelos requisitos apresentados.

2.4. Caracterização dos Perfis de Utilização

Atendendo aos requisitos apresentados, é possível identificar 5 perfis de utilização: Médico, Funcionário, Atleta, Clube e Administrador. Cada um destes perfis tem acesso a diferentes funcionalidades.

O Administrador serve para identificar a Administração da Clínica+, isto é, os 4 amigos descritos na Contextualização. Estes tem todas as permissões na base de dados, ou seja, pode fazer alterações diretas no sistema no que toca à atualização, inserção e consulta de dados.

O Médico pode consultar o histórico e agenda de testes. Para além disto, tem acesso às *procedures* relativas aos dados do Atleta e aos resultados dos testes.

O Funcionário é responsável pela gestão da agenda dos testes, ou seja, pode cancelar, marcar, alterar testes. Também pode consultar os Atletas.

O Atleta está sempre associado a um Clube, e estes tem permissões parecidas. Ambos podem alterar e consultar dados relativos ao Atleta. No entanto, o Clube pode inserir atletas e adicionar categoria a um atleta.

3. Modelação Conceptual

3.1. Identificação e caracterização das entidades

De modo a realizar a elaboração do modelo conceptual começamos primeiramente por identificar as entidades do problema. Para isto é crucial uma observação minuciosa dos requisitos, para não haver uma seleção incorreta destes, pois, por vezes, existem aspetos com importância para o modelo, mas que não são necessariamente uma entidade.

Assim sendo, iremos demonstrar as entidades escolhidas acompanhadas com o seu significado.

Entidade	Descrição
Atleta	Descreve todos os utilizadores da clínica registados na base de dados.
Médico	Representa todos os médicos da clínica que estão registados no sistema.
Especialidades	Descreve a especialidade de um médico, e implicitamente a especialidade do teste.
Turno	Descreve o turno no qual o médico trabalha.
Categoria	Descreve qual a categoria da modalidade que o atleta pratica.
Modalidade	Informar qual a modalidade em que o atleta está inscrito.
Testes	Representa todos os testes realizados.

Tabela 1 - Entidades

3.2. Identificação e caracterização dos relacionamentos

No seguimento da construção do modelo conceptual, procedeu-se à identificação dos relacionamentos existentes entre as entidades previamente apontadas. Tivemos novamente a necessidade de recorrer aos requisitos registados para os estabelecer. Temos então os seguintes relacionamentos:

Atleta-Teste:

Este relacionamento corresponde à realização de um teste clínico por parte da entidade Atleta. Cada Atleta pode realizar vários testes diferentes, por isso estamos perante um relacionamento 1 para N com participação opcional por parte do atleta e obrigatória do lado do teste.

Atleta-Modalidade/Categoria:

Uma modalidade é definida como uma atividade desportiva associado a atletas, e cada modalidade tem várias categorias. Então, cada atleta possui uma modalidade e uma categoria associada, definindo assim uma relação ternária de N atletas para N modalidade/categoria.

Teste-Médico:

O relacionamento “criado” entre a entidade Teste e a entidade Médico caracteriza-se por uma cardinalidade de N para 1, dado que um Médico pode analisar vários testes e um teste é analisado apenas por um Médico.

Médico-Turno:

Cada turno tem vários médicos a trabalhar, por isso este relacionamento entre as entidades Médico e Turno é de N para 1, pois um Turno pode ter vários Médicos a trabalhar e um Médico trabalha apenas num Turno.

Médico-Especialidade:

Por fim, temos o relacionamento Médico-Especialidade que define a especialidade, a nível profissional, de cada médico. Estamos perante um relacionamento de N para 1, porque cada Médico tem a sua Especialidade mas há vários Médicos com a mesma Especialidade.

3.2.1 Dicionário de relacionamento

Entidade	Cardinalidade	Relacionamento	Cardinalidade	Entidade
Atleta	1 N	Realiza Compete	N N	Teste Categoria/Modalidade
Teste	N	Analisado por	1	Médico
Médico	N N	Responsável Trabalha	N 1	Especialidade Turno

Tabela 2 - Dicionário de Relacionamento

3.3. Identificação e caracterização das Associações dos Atributos com as Entidades e Relacionamentos

Entidade/Relacionamento	Atributos	Descrição	Tipo de Atributo	Tipos de dados e tamanho
Atleta (Entidade)	**id_atleta**	Identificador do atleta	Chave primária	INT
	Morada	Morada do atleta	Simples	VARCHAR(128)
	Clube	Clube do atleta	Simples	VARCHAR(64)
	Nome	Nome do atleta	Simples	VARCHAR(64)
	Data de nascimento	Data de nascimento do atleta	Simples	DATE
	Contacto	Contacto do atleta	Simples	VARCHAR(64)
	Apto	Aptidão física	Simples	VARCHAR(8)
	Nacionalidade	Nacionalidade do Atleta	Simples	VARCHAR(32)
Teste (Entidade)	**id_teste**	Identificador do teste	Chave primária	INT
	Estado	Estado do teste	Simples	VARCHAR(16)

	Agendado	Data do agendamento	Simples	DATE
Médico (Entidade)	**id_médico**	Identificador do médico	Chave primária	INT
	Nome	Nome do médico	Simples	VARCHAR(64)
Especialidade (Entidade)	**id_especialidade**	Identificador da especialidade	Chave primária	INT
	Preço	Custo da formação	Simples	DECIMAL
	Nome	Nome da especialização	Simples	VARCHAR(32)
Turno (Entidade)	**id_turno**	Identificador do turno	Chave primária	INT
	Nome	Nome do turno	Simples	VARCHAR(32)
	Inicio	Início do turno	Simples	TIME
	Fim	Fim do turno	Simples	TIME
Categoria (Entidade)	**id_categoria**	Identificador da categoria	Chave primária	INT
	Nome	Nome da categoria	Simples	VARCHAR(32)
Modalidade (Entidade)	**id_modalidade**	Identificador da modalidade	Chave primária	INT
	Nome	Nome da modalidade	Simples	VARCHAR(32)
Contém (Relacionamento)				

Tabela 3 - Dicionário de dados dos atributos das entidades e relacionamentos

3.4. Determinação das chaves candidatas, chaves primárias e chaves alternadas

Agora que identificamos as entidades, os seus atributos e os relacionamentos estabelecidos, podemos verificar quais as chaves primárias de cada entidade. É importante referir que uma chave primária corresponde a um atributo, que identifica a identidade inequivocamente.

As chaves candidatas são todos os atributos que caracterizam a entidade e a torna "única". As chaves não primárias denominam-se chaves alternadas. Assim, serão apresentadas de seguida as chaves de cada entidade do Modelo Conceptual.

Atleta

Chaves candidatas: Contacto, id_atleta

Chave primária: id_atleta

Teste

Chaves candidatas: id_teste

Chave primária: id_teste

Médico

Chaves candidatas: id_médico

Chave primária: id_médico

Especialidade

Chaves candidatas: Nome, id_especialidade

Chave primária: id_especialidade

Turno

Chaves candidatas: Nome, id_turno

Chave primária: id_turno

Categoria

Chaves candidatas: Nome, id_categoria

Chave primária: id_categoria

Modalidade

Chaves candidatas: Nome, id_modalidade

Chave primária: id_modalidade

3.5. Detalhe ou generalização de entidades

Uma vez que as entidades apresentadas representam objetos distintos com diferentes papéis no decorrer da atividade da clínica, não houve a necessidade de proceder ao detalhe ou generalização das entidades.

Como não há entidades diferentes com atributos idênticos, não há necessidade de utilizar generalizações.

Também não é necessário proceder ao detalhe das entidades, pois as ocorrências das entidades de um mesmo tipo terão características semelhantes e acontecerão sobre contextos idênticos.

3.6. Apresentação e explicação do diagrama ER

Tendo definido as entidades, relacionamentos e atributos, temos o seguinte diagrama ER:

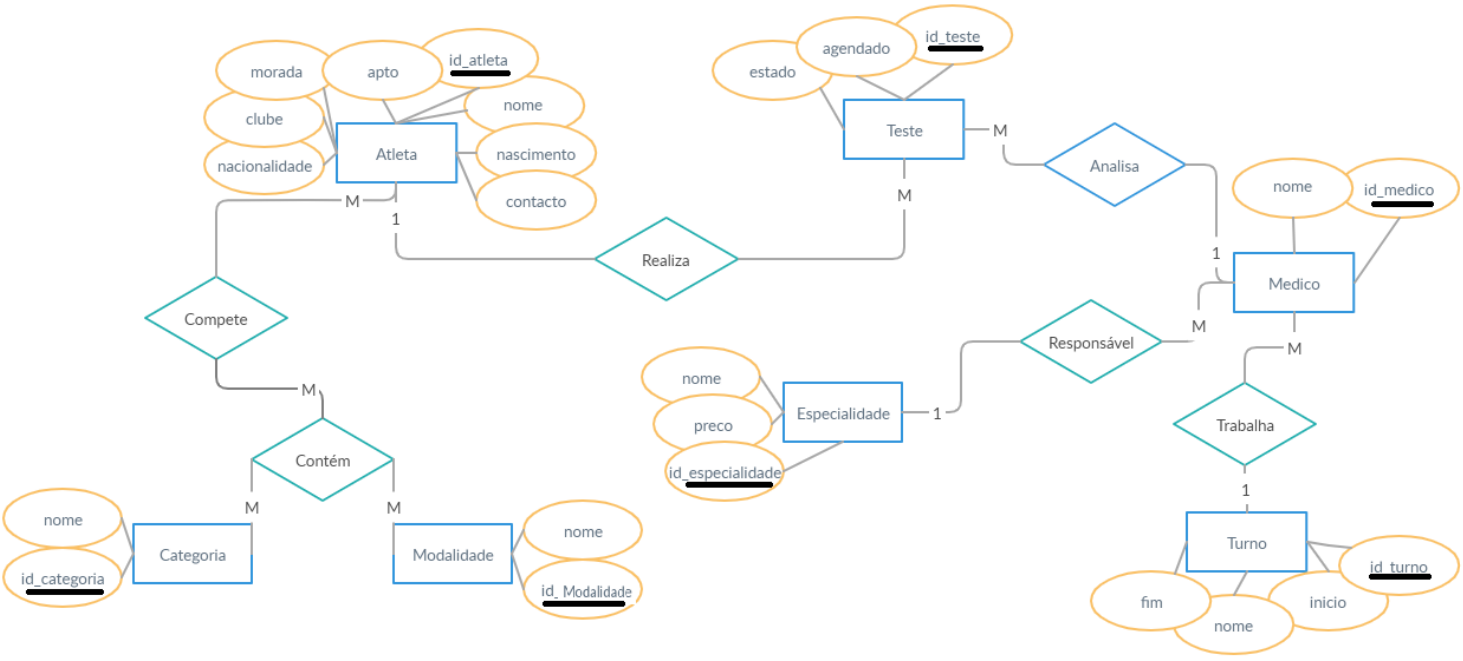

Figura 1 - Diagrama ER

Analisando o diagrama apresentado, constatamos que as entidades e os atributos que lhe estão associados se encontram em conformidade com os requisitos de exploração enunciados. Também os relacionamentos estão bem refletidos conforme o anteriormente dito. A justificação dos relacionamentos e respetivas multiplicidades encontra-se na secção 3.2.

3.7. Validação do modelo de dados com o utilizador

Uma vez terminado o Modelo Conceptual, vamos agora validar o mesmo. Para tal, é fundamental ser possível responder a todas as perguntas que o utilizador possa elaborar.

Então, escolhemos as questões posteriormente escritas, de modo a verificar a viabilidade com o Modelo Conceptual.

1. Quais são os próximos testes de um atleta? (RE1)

Para obter a informação necessária para a resposta, é fundamental recorrer às entidades Atleta e Teste. Tendo em conta o relacionamento entre o Atleta e Teste é possível ter acesso a todos os testes agendados por um Atleta.

2. Quais são os médicos disponíveis por turno? (RE2)

Serão necessárias apenas as entidades Médico e Turno, porque existe um relacionamento direto entre estas entidades. Então é possível saber quantos médicos há por turno dado esta relação.

3. Obter os resultados de um teste? (RE3)

Através da relação direta Atleta-Teste, não há a necessidade de outras entidades e de outras relações. Então é possível obter os resultados de um teste apenas com este relacionamento.

4. Quantos testes foram realizados por um atleta? (RE4)

Para obter a informação necessária para a resposta, é fundamental recorrer às entidades Atleta e Teste. Tendo em conta o relacionamento entre o Atleta e Teste é possível ter acesso a todos os testes realizado por um Atleta.

5. Testes de uma certa especialidade realizados por um atleta? (RE5)

Como não há uma relação direta entre o Atleta e a Especialidade, é preciso também a entidade Médico e Teste. Através da relação Especialidade-Médico sabemos quantos Médicos especializaram-se nessa Especialidade. Desses Médicos, determinamos os testes realizados por cada Médico através do relacionamento entre Médico e Teste. Através do relacionamento Teste-Atleta sabemos quantos atletas realizaram esse Teste com a Especialidade indicada.

4. Modelação Lógica

Após o Modelo Conceptual ter sido validado pelo cliente, prosseguimos à definição do Modelo Lógico do sistema. A sua construção foi orientada pelo modelo de dados relacional. Segue-se neste capítulo o processo de construção do modelo, sendo o resultado final apresentado de seguida.

Esta será uma etapa fundamental para o desenvolvimento do novo sistema de gestão da base de dados uma vez que nos permitirá derivar os relacionamentos. Este modelo irá suportar o nosso problema.

4.1. Construção e validação do modelo de dados lógico

Começamos por tentar transpor o Modelo Conceptual, as suas entidades, relacionamentos e atributos, de modo a elaborar o Modelo Lógico. Após uma análise faseada do Modelo Conceptual, em cada etapa identificamos ocorrências de certos elementos no modelo, derivando de imediato a sua representação no modelo lógico.

A ordem de identificação foi a seguinte:

- Entidades fortes
- Entidades fracas
- Relacionamentos binários de um-para-muitos
- Relacionamentos de um-para-um
- Relacionamentos recursivos de um-para-um
- Relacionamentos do tipo superclasse/subclasse
- Relacionamentos de muitos-para-muitos
- Relacionamentos complexos
- Atributos multivalorados

Relativamente à caracterização das entidades, concluímos que todas as entidades do Modelo Conceptual são entidades fortes. Uma entidade forte define-se por possuir chave primária, que a identifica e não depende de outras entidades

Assim, para cada uma delas, criamos uma relação que inclui todos os atributos simples dessa entidade, dando relevo à chave primária e tendo em conta as chaves alternativas. Deste modo, temos:

Atleta (id_atleta, nome, nascimento, clube, nacionalidade, morada, contacto, apto)

Chave primária: id_atleta

Chaves alternativas: Contacto

Teste (id_testes, agendado, estado)

Chave primária: id_teste

Médico (id_medico, nome)

Chave primária: Id_médico

Especialidade (id_especialidade, nome)

Chave primária: id_especialidade

Chaves alternativas: Nome

Turno (id_turno, nome, inicio, fim)

Chave primária: id_turno

Chaves alternativas: Nome

Categoria (id_categoria, nome)

Chave primária: id_categoria

Chaves alternativas: Nome

Modalidade (id_modalidade, nome)

Chave primária: id_modalidade

Chaves alternativas: Nome

Relativamente aos relacionamentos, identificamos 4 **relacionamentos de um-para-muitos.** Neste tipo de relacionamento, a entidade que apresenta multiplicidade N possui como atributo a chave primária da entidade com multiplicidade 1. Este atributo é chamado de chave estrangeira.

Teste (id_testes, agendado, estado)

Chave primária: id_teste

Chave estrangeira: id_medico, id_atleta

Médico (id_medico, nome)

Chave primária: id_médico

Chave estrangeira: id_especialidade, id_turno

Foram também identificados 2 **relacionamentos de muitos-para-muitos**. Este tipo de multiplicidade gera um novo relacionamento, onde a chave primária composta pelas chaves primárias de cada uma das entidades envolvidas. Neste caso, tivemos em conta a relação Atleta-Modalidade/Categoria e criamos mais 2 relacionamentos no Modelo Lógico.

AtletaCategoria

Chave primária: id_atleta, id_modalidade, id_categoria

ModalidadeCategoria

Chave primária: id_modalidade, id_categoria

Como não identificamos mais nenhuma ocorrência de mais nenhum relacionamento, nem de atributos multivalorados, damos como terminado a parte de derivar e podemos agora representar o Modelo Lógico.

4.2. Desenho do Modelo Lógico

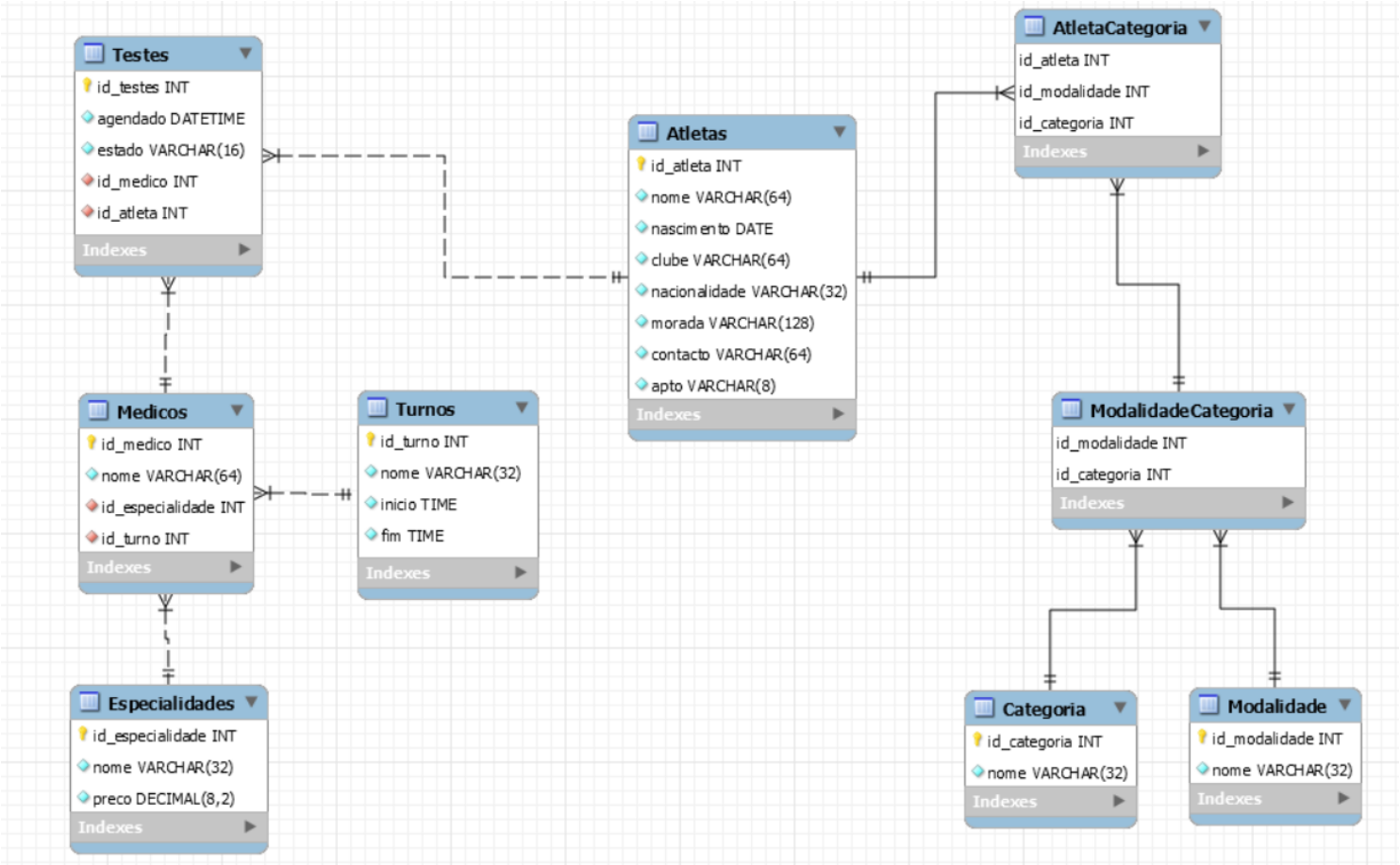

Figura 2 - Modelo Lógico

4.3. Validação do modelo através da normalização

A normalização tem como objetivo assegurar que as relações têm um número mínimo, mas suficiente de atributos necessários para suportar as exigências relativas aos dados a guardar, podemos perceber melhor cada atributo e o que este representa na base de dados. Além disso, permite que a redundância de dados seja mínima possível.

Desta forma, procedemos à verificação do modelo através da normalização, ou seja, verificamos se o modelo está de acordo com a **1ª Forma Normal**, a **2ª Forma Normal** e a **3ª Forma Normal**.

Um relacionamento encontra-se na **1ª Forma Normal** (1FN) se todos os atributos forem atómicos, ou seja, não é possível decompô-los em subconjuntos de atributos e não existirem diferentes atributos que descrevem o mesmo. No nosso modelo conceptual não temos atributos compostos nem multivalorados, por isso não houve a necessidade de decompor qualquer atributo, pelo que eles são todos Simples. Podemos concluir que o nosso modelo está de acordo com a 1FN.

Uma relação está na **2ª Forma Normal** (2FN) se pertencer à 1FN e se todos os atributos não chave dependerem totalmente da chave primária. Como podemos constatar não existe dependência funcional total em relação à chave primária. Assim, o nosso modelo está de acordo com a 2FN.

Um relacionamento encontra-se na **3ª Forma Normal** (3FN) se está na 2FN e se todos os atributos não chave só dependerem unicamente da chave primária e não dependerem de um outro atributo que por sua vez dependesse da chave primária. Como nenhum atributo de nenhuma relação depende de outro atributo que não seja a chave primária, o modelo respeita a 3FN.

4.4. Validação do modelo com interrogações do utilizador

Concluído a normalização do modelo lógico, temos agora que verificar se o modelo que construímos conseguia responder aos requisitos do utilizador e, por isso, recorremos à validação através de interrogações por parte do mesmo.

Assim demonstramos que o modelo lógico consegue responder às questões seguidamente expostas.

1. Quais são os próximos testes de um atleta? (RE1)

Inicialmente, selecionamos da tabela Atletas, o atleta em questão e juntamos o resultado com a tabela Teste. De seguida, filtramos na tabela Testes aqueles que estão com o estado “Marcado”. Por fim, juntamos os resultados, obtendo dessa forma os testes marcados e não realizados dum atleta.

2. Quais são os médicos disponíveis por turno? (RE2)

Filtramos da tabela Turnos o turno em questão, e juntamos o resultado com a tabela Médico. Por fim, projetamos o resultado da junção das tabelas Médicos e Turnos, com a condição Medicos.id_turno = Turnos.id_turno.

3. Qual o resultado de um certo teste? (RE3)

Para responder a esta questão, precisamos da tabela Testes e juntamos a tabela Atletas com a condição Testes.id_atleta = Atleta.id_atleta. Por fim, fazemos seleção do teste com a data certa, através do atributo agendado.

4. Quantos testes foram realizados por um atleta? (RE4)

Inicialmente, selecionamos da tabela Atletas, o atleta em questão e juntamos o resultado com a tabela Testes. De seguida, filtramos na tabela Testes todos aqueles que não têm o estado como “Marcado”. Por fim, juntamos os resultados, obtendo dessa forma os testes marcados e não realizados dum atleta.

5. Testes de uma certa especialidade realizados por um atleta? (RE5)

Para tal, precisamos das tabelas Atletas, Testes, Médicos e Especialidades. Fazemos a junção das tabelas Médicos, Especialidades, Testes e Atletas e filtramos os testes com o estado diferente de “Marcado” e a especialidade escolhida (Especialidade.nome = especialidade). Fazemos apenas a seleção de algumas colunas.

4.5. Validação do modelo com as transações estabelecidas

Uma vez validado o modelo lógico com interrogações do utilizador, chegou o momento de o validar com as transações estabelecidas.

De seguida, iremos apresentar algumas das transações que consideramos importantes para o nosso projeto.

- **Registar Atleta**

  Para adicionar um atleta começa-se por elaborar um novo registo na tabela Atletas, tendo em conta todos os atributos indicados, tal como o contacto, morada, entre outros. Deve ser registado também a modalidade e categoria, efetuando novos registos nas tabelas Modalidade e Categoria, se estas ainda não existirem.

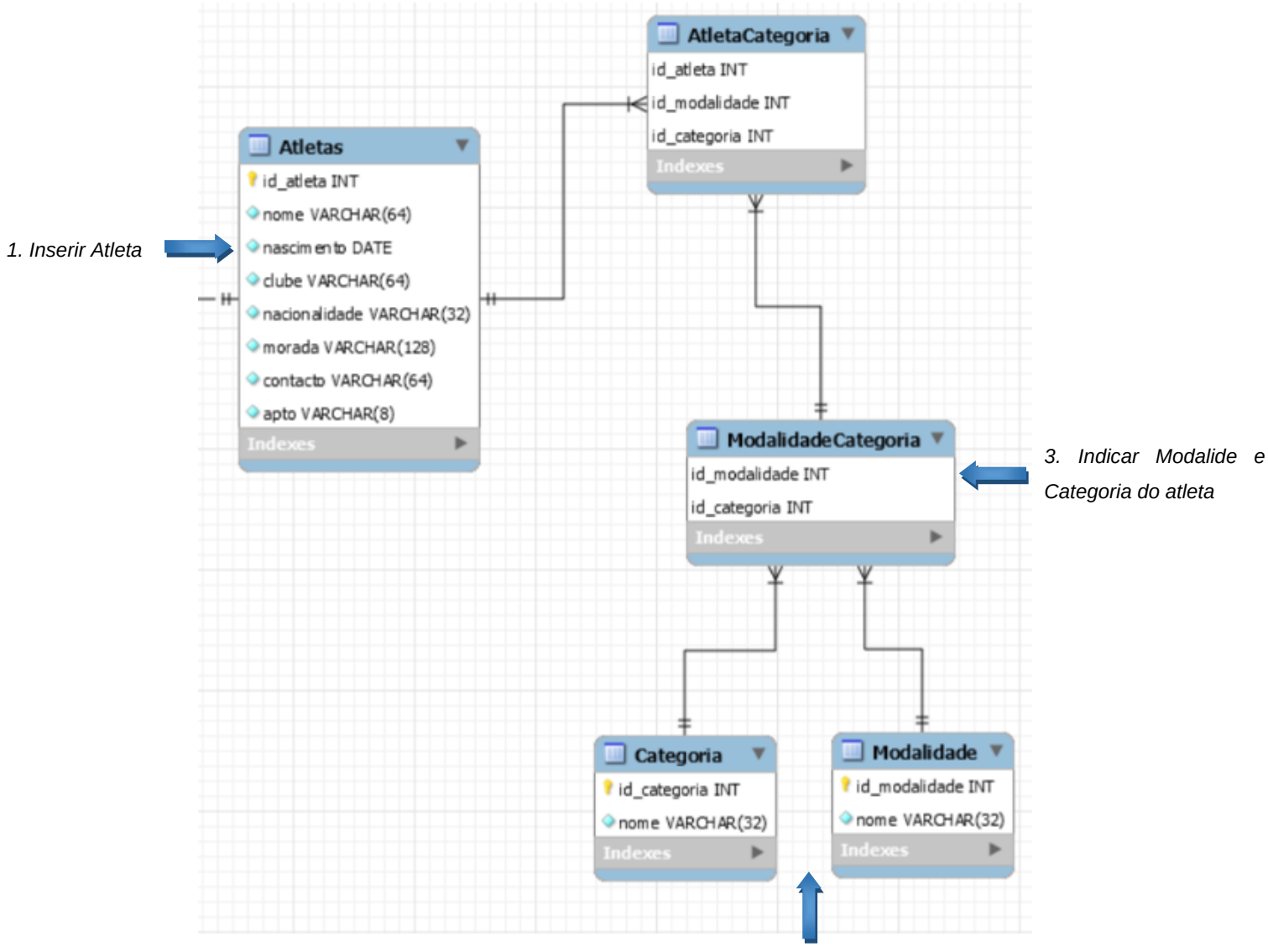

Figura 3 - Registar Atleta

- **Agendar um teste**

  Para adicionar um novo teste, começamos por efetuar o registo do teste fazendo um novo registo na tabela Testes. De seguida, é necessário indicar qual o Atleta que agendou o teste, para que horas agendou esse teste e o seu tipo (especialidade). Para isso, precisamos de saber a especialidade e o turno do médico que vai proceder à execução do teste.

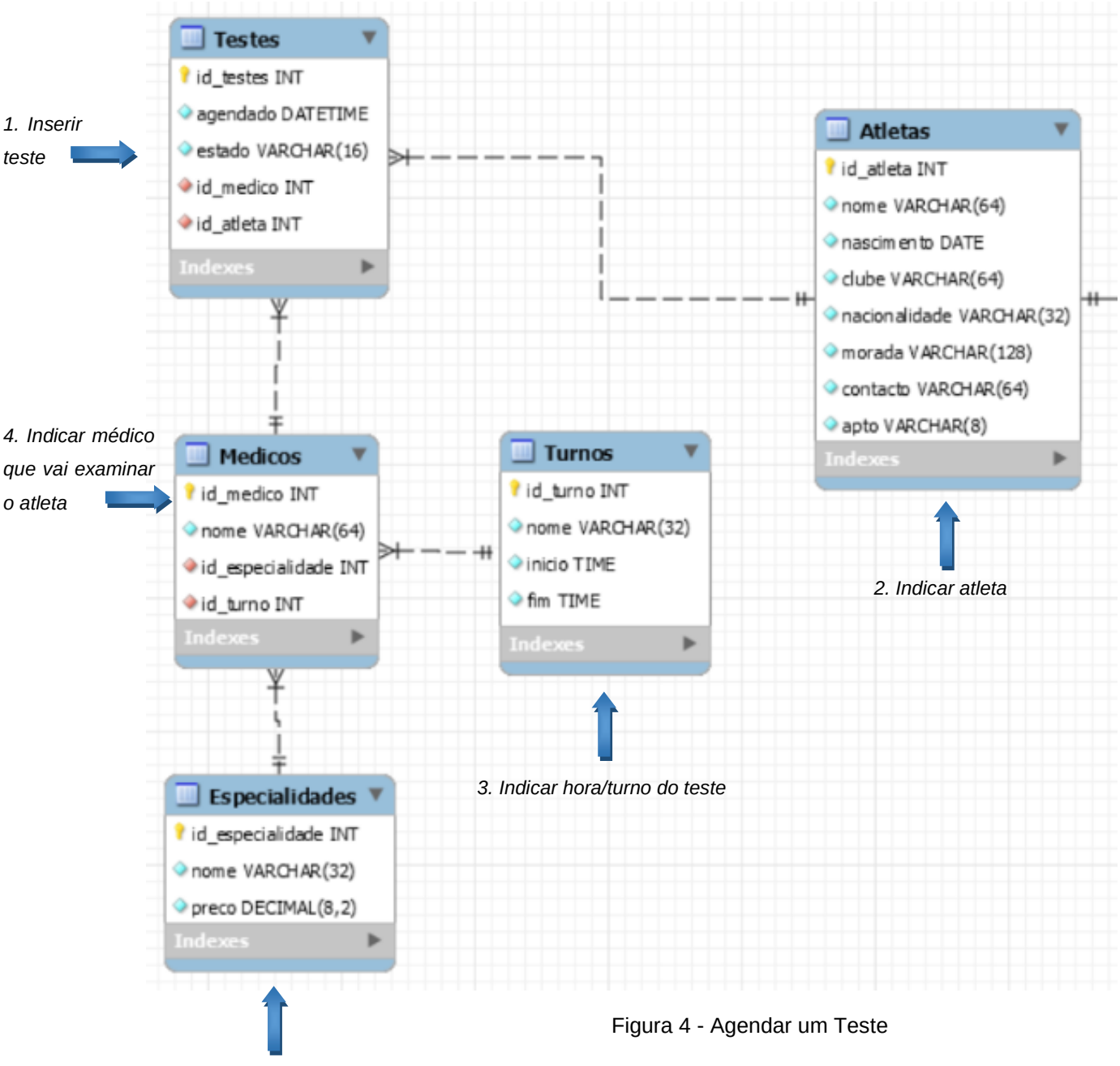

Figura 4 - Agendar um Teste

- **Arquivar/Concluir um teste**

  Para arquivar um teste, quando este fica concluído, é necessário alterar o estado do mesmo na tabela Teste. Este passa de “Marcado” para “Aprovado” ou “Reprovado”, dependendo do resultado do teste.

Concluir o teste e muda o estado do mesmo

Figura 5 - Concluir um Teste

4.6. Revisão do modelo lógico com o utilizador

Tendo como terminado a fase de modelação lógica de base de dados, foi necessário validar este processo com a administração da Clínica+. Para isso, foi marcada mais uma reunião, onde apresentamos e justificamos a estrutura que desenvolvemos.

Por fim, como recebemos a aprovação da administração podemos então começar a elaborar o Modelo Físico.

5. Implementação Física

Como foi referido anteriormente, passamos agora para a fase final do processo de construção do sistema de base dados passou pela sua implementação física, seguida do povoamento, exploração e monitorização da informação armazenada.

Uma vez implementado o esquema físico, procedemos à tradução das interrogações do utilizador e das transações estabelecidas para SQL. Implementamos também mecanismos de segurança com o objetivo de restringir o acesso dos utilizadores a certas partes de base de dados.

Neste capítulo, pretendemos apresentar uma descrição detalhada de cada um dos processos referidos anteriormente e é sumariada a revisão do sistema implementado junto da empresa Clínica+.

5.1. Seleção do sistema de gestão de base de dados

Para o nosso projeto utilizamos como sistema de gestão de base de dados o *MySQL,* pois este é de fácil utilização, e de ter um bom desempenho. Um fator que também influenciou a nossa decisão foi o do *software* ser gratuito. Este disponibiliza mecanismos de controlo de ocorrência, nomeadamente as transações. Permite também a definição de perfis de utilização, útil para delimitar o acesso de cada tipo de utilizador a apenas uma parte da base dados que seja suficiente para satisfazer os seus requisitos.

5.2. Tradução do esquema lógico para o sistema de gestão de base de dados escolhidos em SQL

Para facilitar o processo da implementação física, decidimos usufruir ao máximo dos mecanismos oferecidos pelo *MySQL*. Assim, usamos a ferramenta de desenho, desenvolvimento e administração de base de dados, *MySQL Workbench*. Dado que o nosso modelo lógico foi criado nesta ferramenta, recorremos ao *Forward Engineering* que esta oferece. Este mecanismo permite automatizar o processo top-down de construção de componentes de baixo nível, neste caso a implementação no sistema de gestão de base de dados, através de uma abstração de alto nível, o modelo Lógico. Obtemos assim o código resultante deste processo, e apenas tivemos que o executar para que a base de dados fosse criada.

Segue-se então, o código SQL gerado:

- **Relação Atleta**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`Atletas` (
  `id_atleta` INT NOT NULL,
  `nome` VARCHAR(64) NOT NULL,
  `nascimento` DATE NOT NULL,
  `clube` VARCHAR(64) NOT NULL,
  `nacionalidade` VARCHAR(32) NOT NULL,
  `morada` VARCHAR(128) NOT NULL,
  `contacto` VARCHAR(64) NOT NULL,
  `apto` VARCHAR(8) NOT NULL,
  PRIMARY KEY (`id_atleta`))
ENGINE = InnoDB;
```

Figura 6 - Código SQL da tabela Atleta

- **Relação Teste**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`Testes` (
  `id_testes` INT NOT NULL AUTO_INCREMENT,
  `agendado` DATETIME NOT NULL,
  `estado` VARCHAR(16) NOT NULL,
  `id_medico` INT NOT NULL,
  `id_atleta` INT NOT NULL,
  PRIMARY KEY (`id_testes`),
  INDEX `fk_Testes_Medico1_idx` (`id_medico` ASC) VISIBLE,
  INDEX `fk_Testes_Atletas1_idx` (`id_atleta` ASC) VISIBLE,
  CONSTRAINT `fk_Testes_Medico1`
    FOREIGN KEY (`id_medico`)
    REFERENCES `TestesClinicos`.`Medicos` (`id_medico`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Testes_Atletas1`
    FOREIGN KEY (`id_atleta`)
    REFERENCES `TestesClinicos`.`Atletas` (`id_atleta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 7 - Código SQL da tabela Teste

- **Relação Turno**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`Turnos` (
  `id_turno` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(32) NOT NULL,
  `inicio` TIME NOT NULL,
  `fim` TIME NOT NULL,
  PRIMARY KEY (`id_turno`))
ENGINE = InnoDB;
```

Figura 8 - Código SQL da tabela Truno

- **Relação Médico**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`Medicos` (
  `id_medico` INT NOT NULL,
  `nome` VARCHAR(64) NOT NULL,
  `id_especialidade` INT NOT NULL,
  `id_turno` INT NOT NULL,
  PRIMARY KEY (`id_medico`),
  INDEX `fk_Medico_Especialidades1_idx` (`id_especialidade` ASC) VISIBLE,
  INDEX `fk_Medico_Turnos1_idx` (`id_turno` ASC) VISIBLE,
  CONSTRAINT `fk_Medico_Especialidades1`
    FOREIGN KEY (`id_especialidade`)
    REFERENCES `TestesClinicos`.`Especialidades` (`id_especialidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Medico_Turnos1`
    FOREIGN KEY (`id_turno`)
    REFERENCES `TestesClinicos`.`Turnos` (`id_turno`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 9 - Código SQL da tabela Médico

- **Relação Especialidade**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`Especialidades` (
  `id_especialidade` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(32) NOT NULL,
  `preco` DECIMAL(8,2) NOT NULL,
  PRIMARY KEY (`id_especialidade`))
ENGINE = InnoDB;
```

Figura 10 - Código SQL da tabela Especialidade

- **Relação Modalidade**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`Modalidade` (
  `id_modalidade` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(32) NOT NULL,
  PRIMARY KEY (`id_modalidade`))
ENGINE = InnoDB;
```

Figura 11 - Código SQL para tabela Modalidade

- **Relação Categoria**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`Categoria` (
  `id_categoria` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(32) NOT NULL,
  PRIMARY KEY (`id_categoria`))
ENGINE = InnoDB;
```

Figura 12 - Código SQL para tabela Categoria

- **Relação ModalidadeCategoria**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`ModalidadeCategoria` (
  `id_modalidade` INT NOT NULL,
  `id_categoria` INT NOT NULL,
  PRIMARY KEY (`id_modalidade`, `id_categoria`),
  INDEX `fk_Modalidade_has_Categoria_Categoria1_idx` (`id_categoria` ASC) VISIBLE,
  INDEX `fk_Modalidade_has_Categoria_Modalidade1_idx` (`id_modalidade` ASC) VISIBLE,
  CONSTRAINT `fk_Modalidade_has_Categoria_Modalidade1`
    FOREIGN KEY (`id_modalidade`)
    REFERENCES `TestesClinicos`.`Modalidade` (`id_modalidade`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Modalidade_has_Categoria_Categoria1`
    FOREIGN KEY (`id_categoria`)
    REFERENCES `TestesClinicos`.`Categoria` (`id_categoria`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 13 - Código SQL para tabela ModalidadeCategoria

- **Relação AtletaCategoria**

```
CREATE TABLE IF NOT EXISTS `TestesClinicos`.`AtletaCategoria` (
  `id_atleta` INT NOT NULL,
  `id_modalidade` INT NOT NULL,
  `id_categoria` INT NOT NULL,
  PRIMARY KEY (`id_atleta`, `id_modalidade`, `id_categoria`),
  INDEX `fk_Atletas_has_ModalidadeCategoria_ModalidadeCategoria1_idx` (`id_modalidade` ASC, `id_categoria` ASC) VISIBLE,
  INDEX `fk_Atletas_has_ModalidadeCategoria_Atletas1_idx` (`id_atleta` ASC) VISIBLE,
  CONSTRAINT `fk_Atletas_has_ModalidadeCategoria_Atletas1`
    FOREIGN KEY (`id_atleta`)
    REFERENCES `TestesClinicos`.`Atletas` (`id_atleta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Atletas_has_ModalidadeCategoria_ModalidadeCategoria1`
    FOREIGN KEY (`id_modalidade` , `id_categoria`)
    REFERENCES `TestesClinicos`.`ModalidadeCategoria` (`id_modalidade` , `id_categoria`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 14 - Código SQL para tabela AtletaCategoria

Após a identificação das relações base procedemos à representação dos atributos derivados definindo a forma de os calcular.

Em conformidade com os requisitos levantados, temos que o atleta só fica apto se o último teste de cada especialidade tiver o resultado “Aprovado”. Para isso, criamos 3 *triggers* para atualizar o campo apto em cada atleta consoante a inserção, atualização e cancelamento de testes.

```
DROP TRIGGER IF EXISTS after_update_teste;

DELIMITER $$

CREATE TRIGGER after_update_teste
AFTER UPDATE ON testes
FOR EACH ROW
BEGIN
    DECLARE especialidadesSemApto INT;
    DECLARE atleta INT;

    IF OLD.estado <> NEW.estado THEN

        SET atleta = NEW.id_atleta;
        SET especialidadesSemApto = (SELECT Count(E.id_especialidade) FROM especialidades AS E
                            WHERE E.id_especialidade NOT IN
                                (SELECT M.id_especialidade FROM medicos AS M
                                JOIN testes AS T ON T.id_medico=M.id_medico
                                JOIN (
                                    SELECT M.id_especialidade,max(T.agendado) AS agendado FROM medicos AS M
                                    JOIN testes AS T ON T.id_medico=M.id_medico
                                    WHERE T.id_atleta=atleta
                                    GROUP BY M.id_especialidade
                                     ) AS H
                                ON H.id_especialidade=M.id_especialidade AND H.agendado=T.agendado
                                WHERE T.id_atleta=atleta AND T.estado="Aprovado"));

        IF especialidadesSemApto > 0 THEN
            UPDATE atletas AS A SET A.apto="Nao" WHERE A.id_atleta=atleta;
        ELSE
            UPDATE atletas AS A SET A.apto="Sim" WHERE A.id_atleta=atleta;
        END IF;
    END IF;
END $$
DELIMITER ;
```

Figura 15 - Atualização da aptidão do atleta após realização do teste

5.3. Tradução das interrogações do utilizador para SQL

Nesta secção serão apresentadas as respostas às interrogações do utilizador em código SQL, expressas pelos requisitos de exploração.

- **Verificar as próximas consultas de um atleta (RE1)**

```sql
DROP PROCEDURE IF EXISTS datasTestes;

DELIMITER $$

CREATE PROCEDURE datasTestes (IN atleta INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE existe INT;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    SET existe = (SELECT (Count(A.id_atleta)) FROM atletas AS A WHERE A.id_atleta=atleta);

    SELECT T.id_testes AS "Identificador de teste",T.agendado,M.nome AS "Medico",E.nome AS "Especialidade" FROM testes AS T
    JOIN medicos AS M ON T.id_medico=M.id_medico
    JOIN especialidades AS E ON M.id_especialidade=E.id_especialidade
    WHERE T.id_atleta=atleta AND T.estado="Marcado";

    IF existe=0 THEN
        SET erro=1;
    END IF;

    IF erro THEN
    ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END $$
DELIMITER ;
```

Figura 16 - Código SQL para verificar as próximas consultas de um atleta

- **Ver os médicos disponíveis por turno (RE2)**

```sql
SELECT T.nome AS "Turno", M.nome AS "Médico" FROM turnos AS T
LEFT JOIN medicos AS M ON M.id_turno = T.id_turno
ORDER BY T.id_turno ASC;
```

Figura 17 - Ver os médicos disponíveis por turno

- **Ver os resultados de um teste (RE3)**

```
DROP PROCEDURE IF EXISTS resultadoTeste;

DELIMITER $$

CREATE PROCEDURE resultadoTeste (IN teste INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE existe INT;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    SET existe = (SELECT (Count(T.id_testes)) FROM testes AS T WHERE T.id_testes=teste);

    SELECT T.estado FROM testes AS T WHERE T.id_testes=teste;

    IF existe=0 THEN
        SET erro=1;
    END IF;

    IF erro THEN
    ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END $$
DELIMITER ;
```

Figura 18 - Ver os resultados de um teste

- **Consultar histórico de consultas (RE4)**

```
SELECT T.id_testes AS "Teste", T.agendado AS "Data", T.estado AS "Estado",
M.nome AS "Médico", E.nome AS "Especialidade", A.nome AS "Atleta"
FROM testes AS T
JOIN medicos AS M ON M.id_medico = T.id_medico
JOIN atletas AS A ON A.id_atleta = T.id_atleta
JOIN especialidades AS E ON E.id_especialidade = M.id_especialidade
WHERE NOT T.estado = "Marcado";
```

Figura 19 - Consultar histórico de consultas de um atleta

- **Visualizar todos os testes realizados por especialidade (RE5)**

```
CREATE PROCEDURE testesPorEspecialidade (IN especialidade VARCHAR(32))
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE existe INT;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    SET existe = (SELECT (Count(E.nome)) FROM especialidades AS E WHERE E.nome = especialidade);

    SELECT T.id_testes AS "Teste", T.agendado AS "Data", T.estado AS "Estado",
    M.nome AS "Médico", E.nome AS "Especialidade", A.nome AS "Atleta"
    FROM testes AS T
    JOIN medicos AS M ON M.id_medico = T.id_medico
    JOIN atletas AS A ON A.id_atleta = T.id_atleta
    JOIN especialidades AS E ON E.id_especialidade = M.id_especialidade
    WHERE NOT T.estado = "Marcado" AND E.nome = especialidade;

    IF existe=0 THEN
        SET erro=1;
    END IF;

    IF erro THEN
    ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END $$
DELIMITER ;
```

Figura 20 - Visualizar todos os testes realizados por especialidade

- **Obter o total faturado por mês (RE11)**

```
SELECT monthname(T.agendado) AS "Mês", SUM(E.preco) AS "Total Faturado" FROM testes AS T
JOIN medicos AS M ON M.id_medico = T.id_medico
JOIN especialidades AS E ON E.id_especialidade = M.id_especialidade
WHERE NOT T.estado = "Marcado"
GROUP BY monthname(T.agendado);
```

Figura 21 - Obter o total faturado por mês

- **Verificar a percentagem de aprovação por especialidade (RE12)**

```
SELECT (COUNT(A.apto) / (SELECT COUNT(*) FROM atletas)) * 100 AS "Percentagem de aptos"
FROM atletas AS A
WHERE A.apto = "Sim";
```

Figura 22 - Verificar a percentagem de aprovação por especialidade

- **Obter o número de testes executados por especialidade (RE15)**

```
SELECT E.nome AS "Especialidade", COUNT(*) AS "Número de Consultas" FROM testes AS T
JOIN medicos AS M ON M.id_medico = T.id_medico
JOIN especialidades AS E ON E.id_especialidade = M.id_especialidade
GROUP BY E.nome;
```

Figura 23 - Obter o número de testes executados por especialidade

- **Consultar os próximos testes de um médico (RE19)**

```
DROP PROCEDURE IF EXISTS consultasMedico;

DELIMITER $$

CREATE PROCEDURE consultasMedico (IN medico INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE existe INT;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    SET existe = (SELECT (Count(M.id_medico)) FROM medicos AS M WHERE M.id_medico=medico);

    SELECT M.nome AS "Medico", T.agendado AS "Data", E.nome AS "Especialidade", A.id_atleta AS "CC", A.nome AS "Atleta"
    FROM testes AS T
    INNER JOIN medicos AS M ON M.id_medico=T.id_medico
    INNER JOIN especialidades AS E ON E.id_especialidade=M.id_especialidade
    INNER JOIN atletas AS A ON A.id_atleta=T.id_atleta
    WHERE T.id_medico=medico AND T.estado="Marcado"
    ORDER BY T.agendado ASC,E.nome DESC;

    IF existe=0 THEN
        SET erro=1;
    END IF;

    IF erro THEN
    ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END $$
DELIMITER ;
```

Figura 24 - Consultar os próximos testes de um médico

5.4. Tradução das transações estabelecidas para SQL

Tendo em conta as transações estabelecidas, definimos código em SQL para as executar. De seguida, apresentam-se alguns exemplos de transações.

- **Inserir atleta**

```
CREATE PROCEDURE inserirAtleta(IN id INT, IN nome VARCHAR(64), IN nascimento DATE, IN clube VARCHAR(64), IN nacionalidade VARCHAR(32),
                               IN morada VARCHAR(128), IN contacto VARCHAR(64), IN modalidade VARCHAR(32),
                               IN categoria VARCHAR(32))
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE existe INT DEFAULT 0;
    DECLARE id_modalidade INT;
    DECLARE id_categoria INT;
    DECLARE id_atleta INT;

    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    SET existe = (SELECT Count(A.id_atleta) FROM atletas AS A WHERE A.id_atleta=id);

    IF (existe=0) THEN
        INSERT INTO atletas (id_atleta,nome,nascimento,clube,nacionalidade,morada,contacto,apto) VALUES (id,nome,nascimento,clube,nacionalidade,morada,contacto,"Nao");
    END IF;

    SET id_atleta = (SELECT A.id_atleta FROM atletas AS A WHERE A.id_atleta=id);
    SET id_modalidade = (SELECT M.id_modalidade FROM modalidade AS M WHERE M.nome=modalidade);
    SET id_categoria = (SELECT C.id_categoria FROM categoria AS C WHERE C.nome=categoria);
    SET existe = (SELECT Count(*) FROM modalidadecategoria AS M WHERE M.id_modalidade=id_modalidade AND M.id_categoria=id_categoria);
    INSERT INTO atletacategoria (id_atleta,id_categoria,id_modalidade) VALUES (id_atleta,id_categoria,id_modalidade);

    IF existe=0 OR id_categoria=null OR id_modalidade=null THEN
        SET erro=1;
    END IF;

    IF erro THEN
    ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END $$
DELIMITER ;
```

Figura 25 - Inserir atleta na Clínica+

- **Agendar teste**

```
CREATE PROCEDURE agendarTeste(IN agendado DATETIME,IN especialidade VARCHAR(32),IN atleta INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE existe INT;
    DECLARE existeAtleta INT;
    DECLARE idMedico INT;
    DECLARE id_especialidade INT;
    DECLARE id_teste INT;
    DECLARE dataAtual DATETIME;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    SET dataAtual = (SELECT CURRENT_TIMESTAMP);
    SET id_especialidade = (SELECT E.id_especialidade FROM especialidades AS E WHERE E.nome=especialidade);
    SET existeAtleta = (SELECT Count(A.id_atleta) FROM atletas AS A WHERE A.id_atleta=atleta);
    SET existe = (SELECT Count(T.id_atleta) FROM testes AS T JOIN medicos AS M ON M.id_medico=T.id_medico
                  WHERE T.agendado=agendado AND M.id_especialidade=id_especialidade AND T.id_atleta=atleta);
    SET idMedico = (SELECT M.id_medico FROM medicos AS M JOIN turnos AS TU ON TU.id_turno=M.id_turno
                  WHERE M.id_especialidade=id_especialidade AND (CAST(agendado as TIME) between TU.inicio AND TU.fim)
                  AND M.id_medico NOT IN (SELECT T.id_medico FROM testes AS T WHERE T.agendado=agendado AND M.id_especialidade=id_especialidade)
                  ORDER BY numeroTestes(M.id_medico) ASC
                  LIMIT 1);

    INSERT INTO testes (agendado,estado,id_medico,id_atleta) VALUES (agendado,"Marcado",idMedico,atleta);

     IF id_especialidade = null OR idMedico=null OR existeAtleta=0 OR NOT existe=0 OR dataAtual>agendado OR (SELECT DAYOFWEEK(agendado))=1 THEN
        SET erro=1;
    END IF;

    IF erro THEN
    ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END $$
DELIMITER ;
```

Figura 26 - Agendar/Marcar teste

- **Cancelar um**

```
DROP PROCEDURE IF EXISTS cancelarTeste;

DELIMITER $$

CREATE PROCEDURE cancelarTeste (IN teste INT)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE existe INT;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    SET existe = (SELECT (Count(T.id_testes)) FROM testes AS T WHERE T.id_testes=teste AND T.estado="Marcado");

    DELETE FROM testes AS T WHERE T.id_testes=teste;

    IF existe=0 THEN
        SET erro=1;
    END IF;

    IF erro THEN
    ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END $$
DELIMITER ;
```

Figura 27 - Cancelar teste anteriormente agendado

- **Alterar data de um teste**

```
DELIMITER $$

CREATE PROCEDURE alterarDataTeste (IN teste INT, IN novadata DATETIME)
BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE existe INT;
    DECLARE dataAtual DATETIME;
    DECLARE idMedico INT;
    DECLARE idEspecialidade INT;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;

    START TRANSACTION;

    SET dataAtual = (SELECT CURRENT_TIMESTAMP);
    SET existe = (SELECT (Count(T.id_testes)) FROM testes AS T WHERE T.id_testes=teste AND T.estado="Marcado");
    SET idEspecialidade = (SELECT M.id_especialidade FROM testes AS T JOIN medicos AS M ON T.id_medico=M.id_medico WHERE T.id_testes=teste);
    SET idMedico =  (SELECT M.id_medico FROM medicos AS M JOIN turnos AS TU ON TU.id_turno=M.id_turno
                    WHERE M.id_especialidade=id_especialidade AND (CAST(agendado as TIME) between TU.inicio AND TU.fim)
                    AND M.id_medico NOT IN (SELECT T.id_medico FROM testes AS T WHERE T.agendado=agendado AND M.id_especialidade=id_especialidade)
                    ORDER BY numeroTestes(M.id_medico) ASC
                    LIMIT 1);

    UPDATE testes AS T SET T.agendado=novadata, T.id_medico=idMedico WHERE T.id_testes=teste;

    IF existe=0 OR dataAtual>novadata OR idMedico = null OR idEspecialidade=null OR (SELECT DAYOFWEEK(agendado))=1 THEN
        SET erro=1;
    END IF;

    IF erro THEN
    ROLLBACK;
    ELSE
        COMMIT;
    END IF;

END $$
DELIMITER ;
```

Figura 28 - Alterar data de um teste

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

De seguida, apresenta-se uma estimativa do espaço que a base de dados ocupará em disco. Para precaver qualquer cenário possível, a análise feita ao crescimento da base de dados teve por base a análise do seu pior caso, ou seja, foi assumido que todas as viagens realizadas têm lotação esgotada.

Consideramos os seguintes tamanhos de cada tipo de dados:

- INT -> 4 BYTES
- VARCHAR(N) -> N+2 BYTES
- DATE -> 3 BYTES
- TIME -> 3 BYTES
- DECIMAL -> 5 BYTES

Relação	Atributos	Tipos de dados	Tamanho	TOTAL
Atleta	**id_atleta**	INT	4 bytes	379 bytes
	Morada	VARCHAR(128)	130 bytes	
	Clube	VARCHAR(64)	66 bytes	
	Nome	VARCHAR(64)	66 bytes	
	Data de nascimento	DATE	3 bytes	
	Contacto	VARCHAR(64)	66 bytes	
	Apto	VARCHAR(8)	10 bytes	

	Nacionalidade	VARCHAR(32)	34 bytes	
Teste	**id_teste**	INT	4 bytes	33 bytes
	id_médico	INT	4 bytes	
	id_atleta	INT	4 bytes	
	Estado	VARCHAR(16)	18 bytes	
	Agendado	DATE	3 bytes	
Médico	**id_médico**	INT	4 bytes	78 bytes
	id_especialidade	INT	4 bytes	
	id_turno	INT	4 bytes	
	Nome	VARCHAR(64)	66 bytes	
Especialidade	**id_especialidade**	INT	4 bytes	43 bytes
	Preço	DECIMAL	5 bytes	
	Nome	VARCHAR(32)	34 bytes	

Turno	**id_turno**	INT	4 bytes	44 bytes
	Nome	VARCHAR(32)	34 bytes	
	Inicio	TIME	3 bytes	
	Fim	TIME	3 bytes	
Categoria	**id_categoria**	INT	4 bytes	38 bytes
	Nome	VARCHAR(32)	34 bytes	
ModalidadeCategoria	**id_modalidade**	INT	4 bytes	8 bytes
	id_categoria	INT	4 bytes	
AtletaCategoria	**id_atleta**	INT	4 bytes	12 bytes
	id_modalidade	INT	4 bytes	
	id_categoria	INT	4 bytes	
Modalidade	**id_modalidade**	INT	4 bytes	38 bytes
	Nome	VARCHAR(32)	34 bytes	

Tabela 4 - Espaço ocupado no disco por cada Máquina

Como podemos observar, a entidade Atleta é a que requer maior quantidade de memória.

Agora temos que considerar a dimensão da população na nossa base de dados, em seguida será ilustrado o espaço ocupado no disco por esta.

Tabela	Espaço no disco
Atletas	70*379 = 26530
Médicos	20*78 = 1560
Testes	133*33 = 399
Especialidades	9*43 = 387
Modalidades	4*38 = 152
Categorias	10*38 = 380
ModalidadeCategoria	40*8 = 320
AtletaCategoria	70*12 = 840
Turnos	2*44 = 88
TOTAL	30656

Tabela 5 - Espaço ocupado em disco pela população atual

Como podemos observar na tabela acima, a estimativa para o tamanho da nossa base de dados com o povoamento final é de 30656 bytes. No entanto, é fundamental verificar como se comportará com uma quantidade de dados significante, ou seja, com uma escala real.

Atendendo ao crescimento da clínica nos últimos meses é expectável que dentro de algum tempo estejam inscritos cerca de 200 atletas conduzindo também para maior diversidade de modalidades e categorias. Aumenta também o número de médicos com o crescimento da Clínica+.

5.6. Definição e caracterização das vistas de utilização em SQL

Em SQL, uma view é o resultado de uma ou mais operações relacionais sobre a base de dados. Uma view é uma tabela dinâmica resultante de várias operações entre tabelas ou outras views.

Nesta secção, são apresentadas as *views* que consideramos fundamentais para os utilizadores da nossa base de dados.

```
CREATE VIEW viewAtletas AS
    SELECT A.id_atleta AS "Cartao de Cidadao",A.nome AS "Nome",A.nascimento AS "Data de Nascimento",
           A.nacionalidade AS "Nacionalidade",A.morada AS "Morada", A.contacto AS "Contacto",
           A.clube AS "Clube", M.nome AS "Modalidade",A.apto AS "Apto"
    FROM atletas AS A
    INNER JOIN atletacategoria AS AMC ON AMC.id_atleta=A.id_atleta
    INNER JOIN modalidade AS M ON AMC.id_modalidade=M.id_modalidade
    ORDER BY A.nome ASC;
```

Figura 29 - View Atleta

```
CREATE VIEW viewCategorias AS
    SELECT M.nome AS "Modalidade", C.nome AS "Categoria"
    FROM modalidade AS M
    LEFT JOIN modalidadecategoria AS MC ON MC.id_modalidade=M.id_modalidade
    LEFT JOIN categoria AS C ON MC.id_categoria=C.id_categoria
    ORDER BY M.nome ASC, C.nome ASC;
```

Figura 30 - View Categoria

```sql
CREATE VIEW viewMedicos AS
    SELECT M.nome AS "Nome", E.nome AS "Especialidade",T.nome AS "Turno",T.inicio AS "Inicio", T.fim AS "Fim"
    FROM medicos AS M
    INNER JOIN especialidades AS E ON E.id_especialidade=M.id_especialidade
    INNER JOIN turnos AS T ON T.id_turno=M.id_turno
    ORDER BY M.nome ASC, T.nome ASC;
```

Figura 31 - View Médico

```sql
CREATE VIEW viewTestesAgendados AS
    SELECT T.id_testes AS "Indentificador de teste",T.id_atleta AS "Cartao de Cidadao", A.nome AS "Paciente", T.agendado AS "Data do teste",
           E.nome AS "Especialidade", M.nome AS "Medico",T.estado AS "Estado"
    FROM testes AS T
    INNER JOIN medicos AS M ON M.id_medico=T.id_medico
    INNER JOIN especialidades AS E ON E.id_especialidade=M.id_especialidade
    INNER JOIN atletas AS A ON A.id_atleta=T.id_atleta
    WHERE T.estado="Marcado"
    ORDER BY T.agendado ASC,A.nome ASC;
```

Figura 32 - View Testes Agendados

```sql
CREATE VIEW viewHistoricoTestes AS
    SELECT T.id_testes AS "Indentificador de teste",T.id_atleta AS "Cartao de Cidadao", A.nome AS "Paciente", T.agendado AS "Data do teste",
           E.nome AS "Especialidade", M.nome AS "Medico",T.estado AS "Resultado"
    FROM testes AS T
    INNER JOIN medicos AS M ON M.id_medico=T.id_medico
    INNER JOIN especialidades AS E ON E.id_especialidade=M.id_especialidade
    INNER JOIN atletas AS A ON A.id_atleta=T.id_atleta
    WHERE NOT T.estado="Marcado"
    ORDER BY T.agendado DESC,A.nome ASC;
```

Figura 33 - View Histórico Testes

5.7. Definição e caracterização dos mecanismos de segurança em SQL

Como já foi referido na secção 2.4, foi possível identificar 5 perfis de utilização: Médico, Funcionário, Atleta, Clube e Administrador. Cada um destes perfis tem acesso a diferentes funcionalidades.

De seguida vamos apresentar todos os perfis e as suas permissões.

```
### Administrador
GRANT ALL PRIVILEGES ON * TO 'admin'@'localhost';
```

Figura 34 - Permissões do perfil Administrador

Como podemos ver em cima, o administrador não tem qualquer tipo de restrição podendo, assim, realizar qualquer operação sobre a base de dados ou aceder a qualquer informação nela guardada.

```
### Funcionario
GRANT EXECUTE ON PROCEDURE agendarTeste TO 'funcionario1'@'localhost';
GRANT EXECUTE ON PROCEDURE cancelarTeste TO 'funcionario1'@'localhost';
GRANT EXECUTE ON PROCEDURE alterarDataTeste TO 'funcionario1'@'localhost';
GRANT EXECUTE ON PROCEDURE alterarEspecialidadeTeste TO 'funcionario1'@'localhost';
GRANT SELECT ON viewTestesAgendados TO 'funcionario1'@'localhost';
GRANT SELECT ON viewHistoricoTestes TO 'funcionario1'@'localhost';
GRANT SELECT ON viewatletas TO 'funcionario1'@'localhost';
```

Figura 35 - Permissões do perfil Funcionário

O Funcionário é responsável pela gestão da agenda dos testes, ou seja, pode cancelar, marcar, alterar testes. Também pode consultar os Atletas.

```
### Medico
GRANT EXECUTE ON PROCEDURE concluirTeste TO 'medico1'@'localhost';
GRANT EXECUTE ON PROCEDURE cancelarTeste TO 'medico1'@'localhost';
GRANT EXECUTE ON PROCEDURE resultadosTestes TO 'medico1'@'localhost';
GRANT EXECUTE ON PROCEDURE dadosAtleta TO 'medico1'@'localhost';
GRANT EXECUTE ON PROCEDURE categoriasAtleta TO 'medico1'@'localhost';
GRANT SELECT ON viewTestesAgendados TO 'medico1'@'localhost';
GRANT SELECT ON viewHistoricoTestes TO 'medico1'@'localhost';
```

Figura 36 - Permissões do perfil Médico

O Médico pode consultar o histórico e agenda de testes. Para além disto, tem acesso às *procedures* relativas aos dados do Atleta e aos resultados dos testes.

```
### Atleta
GRANT EXECUTE ON PROCEDURE editarContacto TO 'atleta1'@'localhost';
GRANT EXECUTE ON PROCEDURE editarMorada TO 'atleta1'@'localhost';
GRANT EXECUTE ON PROCEDURE resultadoTeste TO 'atleta1'@'localhost';
GRANT EXECUTE ON PROCEDURE datasTestes TO 'atleta1'@'localhost';
GRANT EXECUTE ON PROCEDURE resultadosTestes TO 'atleta1'@'localhost';
GRANT EXECUTE ON PROCEDURE dadosAtleta TO 'atleta1'@'localhost';
GRANT EXECUTE ON PROCEDURE categoriasAtleta TO 'atleta1'@'localhost';
GRANT SELECT ON viewcategorias TO 'atleta1'@'localhost';
GRANT SELECT ON viewatletas TO 'atleta1'@'localhost';
```

Figura 37 - Permissões do perfil Atleta

```
### Clube
GRANT EXECUTE ON PROCEDURE editarContacto TO 'clube1'@'localhost';
GRANT EXECUTE ON PROCEDURE editarMorada TO 'clube1'@'localhost';
GRANT EXECUTE ON PROCEDURE datasTestes TO 'clube1'@'localhost';
GRANT EXECUTE ON PROCEDURE resultadosTestes TO 'clube1'@'localhost';
GRANT EXECUTE ON PROCEDURE dadosAtleta TO 'clube1'@'localhost';
GRANT EXECUTE ON PROCEDURE categoriasAtleta TO 'clube1'@'localhost';
GRANT EXECUTE ON PROCEDURE inserirAtleta TO 'clube1'@'localhost';
GRANT EXECUTE ON PROCEDURE addCategoriaAtleta TO 'clube1'@'localhost';
GRANT SELECT ON viewcategorias TO 'clube1'@'localhost';
GRANT SELECT ON viewatletas TO 'clube1'@'localhost';
```

Figura 38 - Permissões do perfil Clube

O Atleta está sempre associado a um Clube, e estes tem permissões parecidas. Ambos podem alterar e consultar dados relativos ao Atleta. No entanto, o Clube pode inserir atletas e adicionar categoria a um atleta.

5.8. Revisão do sistema implementado com o utilizador

Uma vez concluída a última etapa da construção do sistema pretendido pela Clínica+, procedemos à apresentação do produto obtido. Para o efeito, reunimos com o corpo administrativo da empresa. Nesse encontro, exploramos as várias funcionalidades desenvolvidas, explicando o seu propósito e procurando exemplifica-las.

O produto final foi aceite com agrado uma vez que cumpria todos os requisitos pretendidos.

6. Abordagem não relacional

6.1. Justificação da utilização de um sistema NoSQL

O sistema de base de dados relacionais assumem uma posição de predominância no mercado. Tal deve-se principalmente à eficácia do armazenamento dos dados estruturados. É importante referir que este tipo de sistemas preservam a consistência, integridade, isolamento dos dados e durabilidade.

Contudo, surgiram problemas que conduziram ao aparecimento de sistema de base de dados não relacionais (*NoSQL*) que foram idealizados atendendo às lacunas que as bases de dados tradicionais demonstraram, com alta performance e capacidade de expansão. Estas em vez de armazenarem os dados em tabelas utilizam estruturas dependendo do tipo da base de dados. Podem ser armazenados em grafos, colunas, chave-valor e documentos.

Dado o elevado número de atletas e tendo em conta todos os testes efetuados por dia, a base de dados cresceu de forma exponencial, sobrecarregando a base de dados relacional e tornando as interrogações mais lentas.

Portanto, após uma reunião administrativa, a alternativa encontrada foi migrar os dados da base relacional que desenvolvemos em SQL para uma baseada em documentos, ou seja, para um modelo *NoSQL*, utilizando para tal **MongoDB.**

6.1.1 MongoDB

Como já foi referido, MongoDB é uma base de dados *NoSQL* baseada em documentos. Suas características permitem que as aplicações modelem informações de modo muito mais natural, pois os dados podem ser agrupados em hierarquias complexas e continuar a ser indexáveis e fáceis de aceder. Para além disto:

- Permitem também uma maior escalabilidade face às relacionais.
- Sendo os requisitos da base de dados na sua maioria de natureza simples, o MongoDB terá uma eficiência maior apresentado os resultados em tempo inferior aos do MySQL.

- Face o enorme crescimento do número de atletas e médicos na clínica, esta pode querer alterar informações dos atletas, como por exemplo adicionar algum novo atributo. Ora, com o SQL não seria possível modificar a base de dados já existente introduzindo novos tipos de dados. Porém, o MongoDB é flexível e dinâmico, ou seja, é permitida a introdução de novas colunas ou campos na base, sem prejudicar os dados já existentes.
- Sendo a base de dados multi-user, ou seja, utilizada por vários utilizadores ao mesmo tempo, o MongoDB apresenta também melhor performance no que toca ao acesso concorrente à base em questão.

Então, dadas estas vantagens associadas ao MongoDB, ficou definido que iriamos proceder à migração dos dados em *MySQL* para uma nova base de dados em MongoDB, ou seja, num modelo não relacional baseada em documentos.

6.2. Identificação e descrição dos objetivos da base de dados

A Clínica+ verificou, então que, tal como inicialmente previsto, o número de atletas representam, cada vez mais, uma maior percentagem no número de elementos do sistema de base de dados atual. Sendo esta a entidade fulcral para o correto funcionamento do sistema, o seu crescimento acentuado é traduzido num aumento do tempo de computação necessário para que um utilizador comum possa tirar proveito do leque de serviços que a Clínica+ oferece.

Com a implementação da base de dados MongoDB, a empresa teceu como seu principal objetivo, atenuar o impacto que o crescimento dos atletas têm na qualidade do serviço prestado aos seus utilizadores. Através do uso de uma base de dados, focada para responder às interrogações necessárias para o uso comum, é possível que o tempo de computação gasto reduza quando comparado ao sistema atual.

6.3. Requisitos de exploração

As questões que serão realizadas sobre o sistema de dados NoSQL incidem sobre os requisitos de exploração apresentados previamente.

Então, temos os seguintes requisitos de exploração:

Do ponto de vista do utilizador/atleta deve ser possível:

1. Verificar as próximas consultas
2. Ver os médicos disponíveis por turno
3. Ver os resultados de um teste
4. Consultar histórico de consultas
5. Visualizar todos os testes realizados por especialidade

Do ponto de vista do administrador:

6. Obter o número total de atletas
7. Obter todos os médicos que trabalham na clínica
8. Obter número de médicos por especialidade
9. Consultar o total faturado desde a abertura da clínica
10. Obter o total faturado por especialidade
11. Obter o total faturado por mês
12. Verificar a percentagem de aprovação por especialidade
13. Verificar a percentagem de aprovação por modalidade e categoria
14. Obter o número de testes executados por médico
15. Obter o número de testes executados por especialidade
16. Obter por cada turno o número de médicos
17. Obter por cada turno o número de testes realizados
18. Obter os resultados de todos os testes realizados
19. Consultar os próximos testes de um médico
20. Consultar os próximos testes de um atleta
21. Verificar os resultados dos atletas de um clube
22. Verificar os próximos testes dos atletas de um clube
23. Apresentar médicos e turnos

A implementação em MongoDB das questões enumeradas serão apresentadas numa das secções que se seguem.

6.4. Estrutura base para o sistema de dados NoSQL

No processo da migração de dados para um sistema de dados MongoDB concluímos que, com base nas interrogações que desejaríamos fazer, a nossa estrutura contemplaria uma coleção: **Testes**.

Testes

Sendo esta a única entidade do sistema, toda a informação dos perfis de utilização referidos anteriormente estão armazenados nesta coleção.

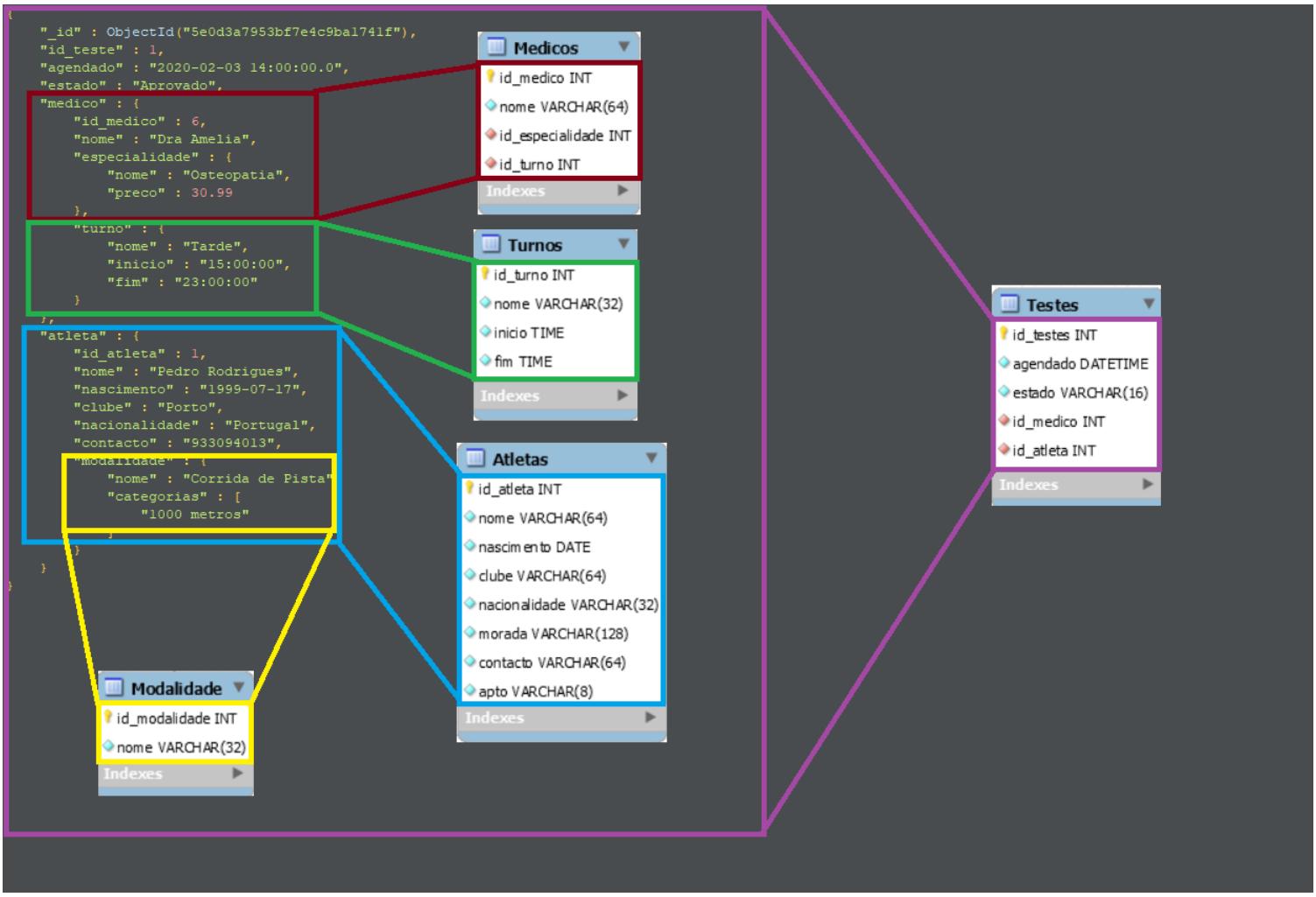

Figura 39 - Construção da Collection Testes

Desta forma, conseguimos resolver tudo sem recorrer a *joins*, porque está tudo na mesma *collection.*

6.5. Identificação de Objetos (SQL) alvo da migração

No caso em concreto, e por forma a ter toda a informação disponível caso seja necessária, todas as tabelas são migradas, sendo que algumas delas não gerarão novas estruturas de dados. Depois de reunida toda a informação, a mesma poderá ser agregada de forma a estruturar da melhor forma o sistema pretendido.

6.6. Migração de dados – Etapas

Extração de dados

Nesta fase a aplicação estabelece a ligação ao servidor SQL, usando o driver mysql, e corremos um comando que nos garante a informação toda necessária.

```
Statement  stm = conn.createStatement();
String query = "SELECT T.id_testes,T.agendado,T.estado,M.id_medico,M.nome, " +
        "E.nome, E.preco,TU.nome, TU.inicio, " +
        "TU.fim, A.id_atleta, A.nome, A.nascimento, A.clube, A.nacionalidade, " +
        "A.morada, A.contacto, MO.nome, C.nome FROM testes AS T " +
        "JOIN medicos AS M ON M.id_medico=T.id_medico " +
        "JOIN especialidades AS E ON E.id_especialidade=M.id_especialidade " +
        "JOIN atletas AS A ON A.id_atleta=T.id_atleta " +
        "JOIN turnos AS TU ON TU.id_turno=M.id_turno " +
        "JOIN atletacategoria AS AC ON A.id_atleta=AC.id_atleta " +
        "JOIN Categoria AS C ON C.id_categoria=AC.id_categoria " +
        "JOIN modalidade AS MO ON MO.id_modalidade=AC.id_modalidade " +
        "ORDER BY T.id_testes";
ResultSet rs = stm.executeQuery(query);
Map<Integer, Testes> testes = new HashMap<>();
```

Figura 40 - Comando SQL

Toda esta informação é guardada num HashMap de Testes, onde a classe Testes tem as seguintes variáveis de instância.

```
private String id_testes;
private String agendado;
private String estado;
private String id_medico;
private String medico;
private String especialidade;
private String preco;
private String turno;
private String turnoEntrada;
private String turnoSaida;
private String id_atleta;
private String nomeAtleta;
private String nascimentoAlteta;
private String clubeAtleta;
private String nacionalidadeAtleta;
private String moradaAtleta;
private String contactoAlteta;
private String modalidade;
private List<String> categoriaAtleta;
```

Figura 41 - Variáveis de instância da classe Testes

Conversão de dados

Nesta fase convertemos esta informação para um ficheiro JSON, com a base de dados pretendida.

```
Testes tt = ttt.get(i);
sb.append("{\n\t\"id_teste\": ");
sb.append(tt.getId_testes()).append(",\n");
sb.append("\t\"agendado\": ");
sb.append("\"" + tt.getAgendado() + "\"").append(",\n");
sb.append("\t\"estado\": ");
sb.append("\"" + tt.getEstado() + "\"").append(",\n");
sb.append("\t\"medico\": {\n");
sb.append("\t\t\"id_medico\": ");
sb.append(tt.getId_medico()).append(",\n");
sb.append("\t\t\"nome\": ");
sb.append("\"" + tt.getMedico() + "\"").append(",\n");
sb.append("\t\t\"especialidade\": {\n");
sb.append("\t\t\t\"nome\": ");
sb.append("\"" + tt.getEspecialidade() + "\"").append(",\n");
sb.append("\t\t\t\"preco\": ");
sb.append(tt.getPreco()).append("\n\t\t},\n");
sb.append("\t\t\"turno\": {\n");
sb.append("\t\t\t\"nome\": ");
sb.append("\"" + tt.getTurno() + "\"").append(",\n");
sb.append("\t\t\t\"inicio\": ");
sb.append("\"" + tt.getTurnoEntrada() + "\"").append(",\n");
sb.append("\t\t\t\"fim\": ");
sb.append("\"" + tt.getTurnoSaida() + "\"").append("\n\t\t}\n\t},\n");
sb.append("\t\"atleta\": {\n");
sb.append("\t\t\"id_atleta\": ");
sb.append(tt.getId_atleta()).append(",\n");
sb.append("\t\t\"nome\": ");
sb.append("\"" + tt.getNomeAtleta() + "\"").append(",\n");
sb.append("\t\t\"nascimento\": ");
sb.append("\"" + tt.getNascimentoAlteta() + "\"").append(",\n");
sb.append("\t\t\"clube\": ");
sb.append("\"" + tt.getClubeAtleta() + "\"").append(",\n");
sb.append("\t\t\"nacionalidade\": ");
sb.append("\"" + tt.getNacionalidadeAtleta() + "\"").append(",\n");
sb.append("\t\t\"contacto\": ");
sb.append("\"" + tt.getContactoAlteta() + "\"").append(",\n");
sb.append("\t\t\"modalidade\": {\n");
sb.append("\t\t\t\"nome\": ");
sb.append("\"" + tt.getModalidade() + "\"");
if (!tt.getCategoriaAtleta().isEmpty()){
    sb.append(",\n\t\t\t\"categorias\": [");
    List<String> cat = tt.getCategoriaAtleta();
    for (int j=0; j<cat.size(); j++){
        sb.append("\"" + cat.get(j) + "\"");
        if (j != cat.size() - 1)
            sb.append(",");
    }
```

Figura 42 - Código de conversão para JSON

Importação de dados

Utilizando o comando ***mongoimport,*** importamos para o *MongoDB server* a partir do ficheiro JSON gerado.

```
String command = "mongoimport --db " + bd + " --collection testes" + " --file " + filePath + " --jsonArray";
System.out.println(command);
processBuilder.command("cmd.exe", "/c", command);
```

Figura 43 - Comando de importação de dados

6.7. Implementação dos Requisitos de Exploração

Nesta secção serão apresentadas as respostas aos requisitos de exploração em código MongoDB, expressas na secção 6.3.

- **Verificar os próximos testes de um atleta (RE1)**

```
db.testes.find({estado: "Marcado"}).pretty();
```

Figura 44 - Verificar os próximos testes de um atleta

- **Ver os médicos disponíveis por turno (RE2)**

```
db.testes.aggregate([
    {
        $group:
        {
            _id: "$medico.turno",
            "medico": {
                $addToSet: "$medico.nome"
            }
        }
    }
]);
```

Figura 45 - Ver os médicos disponíveis por turno

- **Ver os resultados de um teste (RE3)**

```
db.testes.find({id_teste: 5}, {estado: 1});
```

Figura 46 - Ver os resultados de um teste

- **Consultar histórico de consultas de um atleta (RE4)**

```
db.testes.find().pretty();
```

Figura 47 - Consultar histórico de consultas de um atleta

- **Visualizar todos os testes realizados de um atleta por especialidade (RE5)**

```
db.testes.aggregate([
    {
        $match:
        {
            "estado": {$not: {$eq:"Marcado"}}
        }
    },
    {
        $group:
        {
            _id: "$medico.especialidade.nome",
            "Agendado": {$push: "$agendado"},
            "Estado": {$push: "$estado"},
            "Medicos": {$push: "$medico"},
            "Atleta": {$push: "$atleta"}
        }
    }
]);
```

Figura 48 - Visualizar todos os testes realizados de um atleta por especialidade

- **Obter o total faturado por especialidade (RE10)**

```
db.testes.aggregate([
    {
        $match:
        {
            "estado": {$not: {$eq:"Marcado"}}
        }
    },
    {
        $group:
        {
            _id: "$medico.especialidade.nome",
            "TotalAmount": {
                $sum: "$medico.especialidade.preco"
            }
        }
    }
]);
```

Figura 49 - Total faturado por especialidade

- **Verificar a percentagem de aprovação por modalidade e categoria (RE13)**

```
db.testes.aggregate([
    {
        $group:
        {
            _id: "$atleta.modalidade.nome",
            total:{$sum: 1},
            aprovados:{$sum:{$cond:[{ $eq: ["$estado","Aprovado"]},1,0]}}
        }
    },
    {
        $project:
        {
            "percentagem":{$multiply:[{$divide:["$aprovados","$total"]},100]}
        }
    }
]).pretty();
```

Figura 50 - Percentagem de aprovação por modalidade e categoria

- **Obter por cada turno o número de testes realizados (RE17)**

```
db.testes.aggregate([
    {
        $group:
        {
            _id: "$medico.turno.nome",
            "Total de consultas": {$sum:1}
        }
    }
 ]);
```

Figura 51 - Número de testes realizados por turno

- **Verificar os próximos testes dos atletas de um clube (RE22)**

```
db.testes.aggregate([
    {
    $match:{"estado": {$eq:"Marcado"}}
    },
    {
        $group:{
        _id: "$atleta.clube",
        agendado: {$push: "$agendado"},
        atleta: {$push: "$atleta.nome"},
        medico: {$push: "$medico.nome"},
        especialidade: {$push: "$medico.especialidade.nome"},
           }
    }
]).pretty();
```

Figura 52 - Próximos testes dos atletas de um clube

7. Conclusão e notas finais

Fase 1

Além de aplicar os conhecimentos obtidos nas aulas de Base de Dados redirecionados a uma situação, que se tentou o que fosse o mais realística possível, foi possível entender a necessidade de toda a fase de modelação prévia antes da implementação física de uma base de dados

Inicialmente, implementamos o modelo conceptual que permitiu que se extraísse unicamente a informação relevante dos requisitos levantados. Garantimos assim, que a informação relevante era preservada, ajudando na eliminação de redundâncias.

Depois do modelo conceptual, implementamos o modelo lógico que permitiu perceber a melhor forma de estruturar a informação, surgindo novas tabelas de dados, resultantes dos relacionamentos identificados no modelo conceptual e também de atributos.

Com estes modelos, passamos para a implementação física, sendo possível aplicar e aperfeiçoar os conhecimentos da linguagem *MySQL* e ter noção das potencialidades da mesma.

Entendeu-se que o resultado final poderia ser melhorado e tornado mais complexo, e de futuro poder-se-ia fazer uma gestão mais inteligente, por exemplo o uso mais aprofundado de indexes, atributos multivalorados, A nível de perfis de utilização, podíamos ter adicionado o Clube e aumentado a complexidade nos requisitos de controlo.

Fase 2

A migração para um sistema NoSQL, neste caso Mongo permitiu descobrir as potencialidades deste tipo de sistema e a diferente abordagem a ser usada num sistema não relacional. Este novo paradigma implica fortes mudanças na estruturação da base de dados, que é facilmente adaptável ao tipo de uso que se procura obter.

O acesso aos dados tornam alguns casos mais simples, pela agregação que é feita num dado documento de múltiplos tipos de dados sem ter que recorrer a relacionamento de tabelas.

A flexibilidade de dados permite que o sistema seja facilmente adaptável, mas mantendo a consistência dos dados existentes, o que permite soluções menos rígidas ao contrário do que acontecia no paradigma relacional que vão de encontro às necessidades de adaptabilidade dos sistemas hoje em dia.

Em relação à migração efetuada, tentou-se que fosse o mais simples e consistente possível e como explicado anteriormente, usamos comandos como *mongoimport* e ferramentas do *mysql* para a migração dos dados.

No geral entendeu-se que a solução de migração proposta satisfaz todos os requisitos a nível de consistência dos dados. Poderia ser melhorado apenas na parte dos requisitos de controlo, onde podíamos usufruir das *roles*.

Referências

Tratamento Médico de Jogadores – FPF [pdf] Disponível em: https://www.fpf.pt/DownloadDocument.ashx?id=107 [Acedido a 20 Dezembro 2019].

Lista de Siglas e Acrónimos

BD Base de Dados

SGBD Sistema de Gestão de Bases de Dados

DW Data Warehouse

OLTP *On-Line Analytical Processing*

RE Requisito de Exploração

RC Requisito de Controlo

Anexos

<<Os anexos deverão ser utilizados para a inclusão de informação adicional necessária para uma melhor compreensão do relatório o para complementar tópicos, secções ou assuntos abordados. Os anexos criados deverão ser numerados e possuir uma designação. Estes dados permitirão complementar o Índice geral do relatório relativamente à enumeração e apresentação dos diversos anexos.>>

I. Povoamento

```
INSERT INTO `categoria` (`id_categoria`,`nome`) VALUES (1,"100 metros"),
(2,"200 metros"),(3,"400 metros"),(4,"800 metros"),(5,"1000 metros"),
(6,"1500 metros"),(7,"3000 metros"),(8,"5000 metros"),(9,"10000 metros"),(10,"15000 metros");
```

Figura 53 - Povoamento de categorias

```
INSERT INTO `modalidade` (`id_modalidade`,`nome`) VALUES (1,"Marcha"),(2,"Corrida de Pista"),
(3,"Corrida de Estafeta"),(4,"Corrida de Obstaculos");
```

Figura 55 - Povoamento de modalidades

```
INSERT INTO `modalidadecategoria` (`id_modalidade`,`id_categoria`) VALUES (1,1),(1,2),
(1,3),(1,4),(1,5),(1,6),(1,7),(1,8),(1,9),(1,10),(2,1),(2,2),(2,3),(2,4),(2,5),(2,6),
(2,7),(2,8),(2,9),(2,10),(3,1),(3,2),(3,3),(3,4),(3,5),(3,6),(3,7),(3,8),(3,9),(3,10),
(4,1),(4,2),(4,3),(4,4),(4,5),(4,6),(4,7),(4,8),(4,9),(4,10);
```

Figura 54 - Relacionamentos entre modalidade e categoria

```
INSERT INTO `Especialidades` (`id_especialidade`,`nome`,`preco`) VALUES (1,"Cardiologia",50),
(2,"Cardiopulmunar",50),(3,"Análise Sangue",15),(4,"Teste de Urina",15),(5,"HIV",15),
(6,"Ortopedia",35),(7,"Radiologia",20),(8,"Anti-Doping",20),(9,"Dentista",50);
```

Figura 56 - Povoamento de especialidades

```
INSERT INTO `Turnos` (`id_turno`,`nome`,`inicio`,`fim`) VALUES (1,"Manha","06:00","14:00"),(2,"Tarde","14:00","22:00");
```

Figura 57 - Povoamento de Turnos

Para evitar a sobrecarga de figuras, não iremos colocar todos os povoamentos de atletas e testes porque temos um número elevado de linhas, então iremos apenas deixar uns exemplos para representar como fizemos o povoamento.

```
INSERT INTO `Atletas` (`id_atleta`,`nome`,`nascimento`,`clube`,`nacionalidade`,`morada`,`contacto`,`apto`)
VALUES (1,"Daphne Castaneda","2018-02-22","JOMA","Jamaica","389 Ac Avenue","id@congueaaliquet.ca","Nao"),
(2,"Yetta Butler","2018-05-23","Braga","Laos","363-5801 Vitae Road","vestibulum.lorem@placerategetvenenatis.co.uk","Nao"),
(3,"Madaline Benson","2019-10-21","Boavista","Madagascar","P.O. Box 241, 9921 Bibendum Ave","magna@variusultricesmauris.ca","Nao"),
(4,"Liberty Chapman","2019-11-04","Sporting","Malta","Ap #244-8608 Quis Rd.","turpis.vitae@Aliquamultrices.com","Nao"),
(5,"Belle Rivas","2019-08-25","Braga","Djibouti","269-9668 Justo. Rd.","gravida.molestie@vitaepurusgravida.edu","Nao"),
(6,"Logan Floyd","2019-02-28","Sporting","Estonia","Ap #835-5000 Eget Avenue","in@arcuAliquam.net","Nao"),
(7,"Zachery Mayo","2017-11-14","Braga","Dominica","Ap #577-4760 Sapien. Rd.","nisl.Quisque.fringilla@cubiliaCurae.edu","Nao"),
(8,"Fatima Simpson","2017-05-11","Porto","Guadeloupe","3581 Cursus St.","vel.arcu@Mauris.net","Nao"),
(9,"Eve Vinson","2017-06-22","Vidigalense","Wallis and Futuna","Ap #640-7841 Ridiculus Street","enim.nec.tempus@tempusmauris.com","Nao"),
(10,"Kevyn Payne","2017-12-30","Vidigalense","Samoa","924-6797 Tincidunt Street","libero@magnased.co.uk","Nao");
```

Figura 59 - Povoamento de Atletas

```
INSERT INTO `atletacategoria` (`id_atleta`,`id_modalidade`,`id_categoria`)
VALUES (1,4,7),(2,3,10),(3,1,7),(4,1,6),(5,3,4),(6,3,1),(7,1,8),(8,4,7),(9,4,10),(10,1,6);

INSERT INTO `atletacategoria` (`id_atleta`,`id_modalidade`,`id_categoria`)
VALUES (11,4,10),(12,3,7),(13,4,10),(14,1,5),(15,1,2),(16,2,8),(17,3,4),(18,2,10),(19,4,4),(20,3,4);

INSERT INTO `atletacategoria` (`id_atleta`,`id_modalidade`,`id_categoria`)
VALUES (21,2,4),(22,1,10),(23,1,10),(24,3,5),(25,4,2),(26,2,7),(27,2,1),(28,3,1),(29,1,2),(30,3,4);

INSERT INTO `atletacategoria` (`id_atleta`,`id_modalidade`,`id_categoria`)
VALUES (31,2,6),(32,4,2),(33,1,6),(34,4,1),(35,2,9),(36,1,10),(37,1,5),(38,1,1),(39,1,10),(40,2,9);

INSERT INTO `atletacategoria` (`id_atleta`,`id_modalidade`,`id_categoria`)
VALUES (41,4,7),(42,4,9),(43,1,9),(44,4,5),(45,2,6),(46,1,2),(47,1,8),(48,3,8),(49,4,1),(50,2,10);

INSERT INTO `atletacategoria` (`id_atleta`,`id_modalidade`,`id_categoria`)
VALUES (51,2,1),(52,2,6),(53,4,8),(54,1,6),(55,2,7),(56,2,9),(57,3,6),(58,4,2),(59,3,2),(60,3,10);

INSERT INTO `atletacategoria` (`id_atleta`,`id_modalidade`,`id_categoria`)
VALUES (61,3,3),(62,2,1),(63,1,3),(64,4,5),(65,2,10),(66,2,5),(67,1,6),(68,4,6),(69,3,3),(70,3,6);
```

Figura 58 - Relação entre atletas e categorias

Para o povoamento de testes decidimos utilizar o procedure *inserirTeste* para facilitar com a escolha do médico.

```
CALL inserirTeste("2018-11-06 06:29:34","Ortopedia",45);CALL inserirTeste("2017-09-14 07:36:16","Cardiopulmunar",33);
CALL inserirTeste("2019-11-30 09:16:40","Cardiopulmunar",16);CALL inserirTeste("2017-08-26 10:50:35","Ortopedia",67);
CALL inserirTeste("2019-09-11 21:50:48","Dentista",19);CALL inserirTeste("2017-02-13 13:54:38","Radiologia",15);
CALL inserirTeste("2017-08-20 19:59:23","Cardiologia",46);CALL inserirTeste("2019-12-08 09:27:53","Teste de Urina",51);
CALL inserirTeste("2017-11-26 06:39:17","Radiologia",16);CALL inserirTeste("2017-03-21 18:06:56","Cardiologia",57);
CALL inserirTeste("2019-06-24 21:35:32","Teste de Urina",3);CALL inserirTeste("2018-12-12 20:18:01","Dentista",53);
CALL inserirTeste("2018-02-15 17:24:26","Cardiopulmunar",61);CALL inserirTeste("2019-05-22 02:15:39","Anti-Doping",50);
CALL inserirTeste("2019-05-04 06:55:22","Anti-Doping",51);CALL inserirTeste("2019-12-14 21:57:31","Análise Sangue",34);
CALL inserirTeste("2019-01-21 11:09:25","Anti-Doping",25);CALL inserirTeste("2018-10-17 18:10:09","Cardiopulmunar",28);
CALL inserirTeste("2017-05-16 06:14:15","Radiologia",19);CALL inserirTeste("2018-11-07 17:39:37","Análise Sangue",21);
CALL inserirTeste("2019-02-07 17:00:02","HIV",48);CALL inserirTeste("2019-08-31 22:21:33","Teste de Urina",27);
CALL inserirTeste("2018-08-24 11:03:27","Análise Sangue",49);CALL inserirTeste("2019-07-26 03:33:14","Dentista",22);
CALL inserirTeste("2017-04-06 17:55:06","HIV",30);CALL inserirTeste("2017-09-16 05:17:29","Teste de Urina",51);
CALL inserirTeste("2018-01-10 19:42:39","Teste de Urina",23);CALL inserirTeste("2019-03-30 05:39:22","Cardiopulmunar",39);
CALL inserirTeste("2017-11-29 11:35:12","Radiologia",33);CALL inserirTeste("2017-09-18 06:07:26","Ortopedia",41);
CALL inserirTeste("2018-08-02 20:37:33","Radiologia",14);CALL inserirTeste("2018-06-03 11:58:24","Dentista",38);
CALL inserirTeste("2017-11-29 16:57:23","Dentista",34);CALL inserirTeste("2019-04-03 22:15:23","Radiologia",49);
CALL inserirTeste("2019-06-06 18:02:18","Teste de Urina",15);CALL inserirTeste("2019-02-28 21:37:44","Teste de Urina",52);
CALL inserirTeste("2018-08-08 03:15:26","Análise Sangue",68);CALL inserirTeste("2019-01-28 18:15:26","Anti-Doping",22);
CALL inserirTeste("2018-08-29 02:07:51","Cardiopulmunar",61);CALL inserirTeste("2019-01-15 03:44:41","Ortopedia",2);
CALL inserirTeste("2017-12-10 14:10:29","Análise Sangue",42);CALL inserirTeste("2017-01-20 07:03:31","HIV",22);
CALL inserirTeste("2018-07-11 11:09:39","HIV",20);CALL inserirTeste("2017-04-07 04:07:29","Anti-Doping",24);
CALL inserirTeste("2018-07-04 07:34:20","Ortopedia",43);CALL inserirTeste("2017-02-22 22:39:14","Análise Sangue",41);
CALL inserirTeste("2017-04-22 18:16:07","HIV",34);CALL inserirTeste("2019-07-08 23:47:56","Radiologia",45);
CALL inserirTeste("2019-06-16 21:18:28","Anti-Doping",3);CALL inserirTeste("2017-07-31 16:30:38","Cardiopulmunar",37);
CALL inserirTeste("2019-01-09 19:25:08","Análise Sangue",26);CALL inserirTeste("2019-06-05 08:34:45","Dentista",11);
CALL inserirTeste("2018-07-08 20:00:43","Radiologia",52);CALL inserirTeste("2019-03-13 05:59:47","Cardiopulmunar",59);
CALL inserirTeste("2017-10-17 19:13:23","Radiologia",18);CALL inserirTeste("2017-07-22 13:35:49","Teste de Urina",13);
CALL inserirTeste("2017-04-25 12:15:46","Cardiologia",14);CALL inserirTeste("2018-12-16 14:55:22","Anti-Doping",1);
CALL inserirTeste("2017-06-01 10:30:58","Radiologia",28);CALL inserirTeste("2018-08-29 07:13:41","Cardiologia",32);
```

Figura 60 - Povoamento de Testes

Como os testes foram povoados utilizando o procedure, tiveram de ser agora concluídos.

```
CALL concluirTeste(1,"Aprovado");CALL concluirTeste(2,"Aprovado");CALL concluirTeste(3,"Reprovado");
CALL concluirTeste(4,"Aprovado");CALL concluirTeste(5,"Aprovado");CALL concluirTeste(6,"Aprovado");
CALL concluirTeste(7,"Aprovado");CALL concluirTeste(8,"Aprovado");CALL concluirTeste(9,"Aprovado");
CALL concluirTeste(10,"Aprovado");CALL concluirTeste(11,"Aprovado");CALL concluirTeste(12,"Aprovado");
CALL concluirTeste(13,"Aprovado");CALL concluirTeste(14,"Aprovado");CALL concluirTeste(15,"Aprovado");
CALL concluirTeste(16,"Aprovado");CALL concluirTeste(17,"Aprovado");CALL concluirTeste(18,"Aprovado");
CALL concluirTeste(19,"Aprovado");CALL concluirTeste(20,"Aprovado");CALL concluirTeste(21,"Aprovado");
CALL concluirTeste(22,"Aprovado");CALL concluirTeste(23,"Aprovado");CALL concluirTeste(24,"Aprovado");
CALL concluirTeste(25,"Reprovado");CALL concluirTeste(26,"Aprovado");CALL concluirTeste(27,"Aprovado");
CALL concluirTeste(28,"Aprovado");CALL concluirTeste(29,"Aprovado");CALL concluirTeste(30,"Aprovado");
CALL concluirTeste(31,"Reprovado");CALL concluirTeste(32,"Aprovado");CALL concluirTeste(33,"Aprovado");
CALL concluirTeste(34,"Aprovado");CALL concluirTeste(35,"Aprovado");CALL concluirTeste(36,"Aprovado");
CALL concluirTeste(37,"Aprovado");CALL concluirTeste(38,"Aprovado");CALL concluirTeste(39,"Aprovado");
CALL concluirTeste(40,"Aprovado");CALL concluirTeste(41,"Aprovado");CALL concluirTeste(42,"Aprovado");
CALL concluirTeste(43,"Aprovado");CALL concluirTeste(44,"Aprovado");CALL concluirTeste(45,"Aprovado");
CALL concluirTeste(46,"Aprovado");CALL concluirTeste(47,"Reprovado");CALL concluirTeste(48,"Aprovado");
CALL concluirTeste(49,"Aprovado");CALL concluirTeste(50,"Aprovado");CALL concluirTeste(51,"Aprovado");
CALL concluirTeste(52,"Aprovado");CALL concluirTeste(53,"Aprovado");CALL concluirTeste(54,"Aprovado");
CALL concluirTeste(55,"Aprovado");CALL concluirTeste(56,"Aprovado");CALL concluirTeste(57,"Aprovado");
CALL concluirTeste(58,"Aprovado");CALL concluirTeste(59,"Aprovado");CALL concluirTeste(60,"Aprovado");
CALL concluirTeste(61,"Aprovado");CALL concluirTeste(62,"Aprovado");CALL concluirTeste(63,"Reprovado");
CALL concluirTeste(64,"Aprovado");CALL concluirTeste(65,"Aprovado");CALL concluirTeste(66,"Aprovado");
CALL concluirTeste(67,"Aprovado");CALL concluirTeste(68,"Aprovado");CALL concluirTeste(69,"Aprovado");
CALL concluirTeste(70,"Aprovado");CALL concluirTeste(71,"Aprovado");CALL concluirTeste(72,"Aprovado");
```

Figura 61 - Conclusão dos Testes Povoados