



Universidade do Porto

**FEUP** Faculdade de Engenharia

# Relatório de Projeto

## Tema: Hospital

Unidade Curricular: Bases de Dados

Professor(a) Regente: Carla Teixeira Lopes

Professor(a) das Teórico-Práticas: João Mendes Moreira

Ana Margarida Ruivo Loureiro

Carlos Jorge Direito Albuquerque

Gonçalo Santos Oliveira

Turma 7 Grupo 02

# Índice

<b>Introdução</b>	<b>2</b>
<b>Contextualização</b>	<b>3</b>
<b>Modelo Conceptual</b>	<b>4</b>
Diagrama UML	5
Considerações	5
Diagrama UML Revisto	7
Dependências funcionais	9
Formas Normais	10
Modelo Normalizado	12
<b>Implementação da base de dados</b>	<b>13</b>
Restrições	13
Interrogações	16
Triggers	17
1º Trigger	17
2ºTrigger	17
3ºTrigger	18
<b>Conclusão</b>	<b>19</b>

# Introdução

Este projeto tem como objetivos a criação e interrogação de uma base de dados no contexto de um hospital, por forma a auxiliar na organização dos seus dados e tornar os seus serviços mais eficientes.

Assim, iremos desenvolver uma base de dados para que possam ser guardadas informações acerca de todos os processos que estão a decorrer no hospital, os intervenientes nesses processos e no serviço hospitalar em geral, assim como dados sobre medicações e tratamentos e os horários de todos os funcionários.

# Contextualização

A base de dados que iremos construir irá permitir o correto armazenamento dos dados de um hospital de modo a que seja possível o acesso às seguintes informações:

- **Pessoas** que estão associadas ao hospital e as suas informações básicas, das quais se distinguem **utentes** e **funcionários** (**médicos** e **enfermeiros**);
- **Departamentos** em que o hospital se encontra dividido e onde estão repartidos os médicos e enfermeiros pelas áreas de trabalho;
- **Processos** associados a cada entrada dada por um utente no hospital, podendo esta ser relativa a uma consulta, urgência ou internamento e que estarão entregues a médicos e enfermeiros responsáveis;
- **Tratamentos** resultantes da entrada dada no hospital pelo utente;
- **Medicamento** que estará associado ao processo através de uma **prescrição** médica;
- **Alergias** associadas a um paciente através de um **grau de intolerância**;
- **Agenda** que estará associada aos enfermeiros e médicos por forma a registar os seus horários diariamente.

# Modelo Conceptual

Passamos agora a uma análise ao modelo conceptual definido para este projeto.

Como já foi referido, os indivíduos que constituem o hospital vão ser representados pela classe **Pessoa** que será dividida em mais duas subclasses, **Utente** e **Funcionário**, que irão possuir várias relações com todas as restantes classes da base de dados. É também de referir que a classe Funcionário será especializada em outras duas: **Enfermeiro** e **Médico**. Em relação a estas classes é de ter em atenção que um médico tem sempre especialidade, enquanto um enfermeiro pode ou não ter.

De modo a garantir a coesão de horários, cada funcionário tem uma **Agenda**. Através da data em que começou o horário, é possível saber quais as horas de entrada e de saída. Assim, garante-se que todos os turnos são registados na base de dados.

No hospital existem vários departamentos (classe **Departamento**), sendo identificados pelo seu código de localização único. É de notar que em cada departamento operam vários funcionários e cada um destes pode trabalhar em mais do que um departamento.

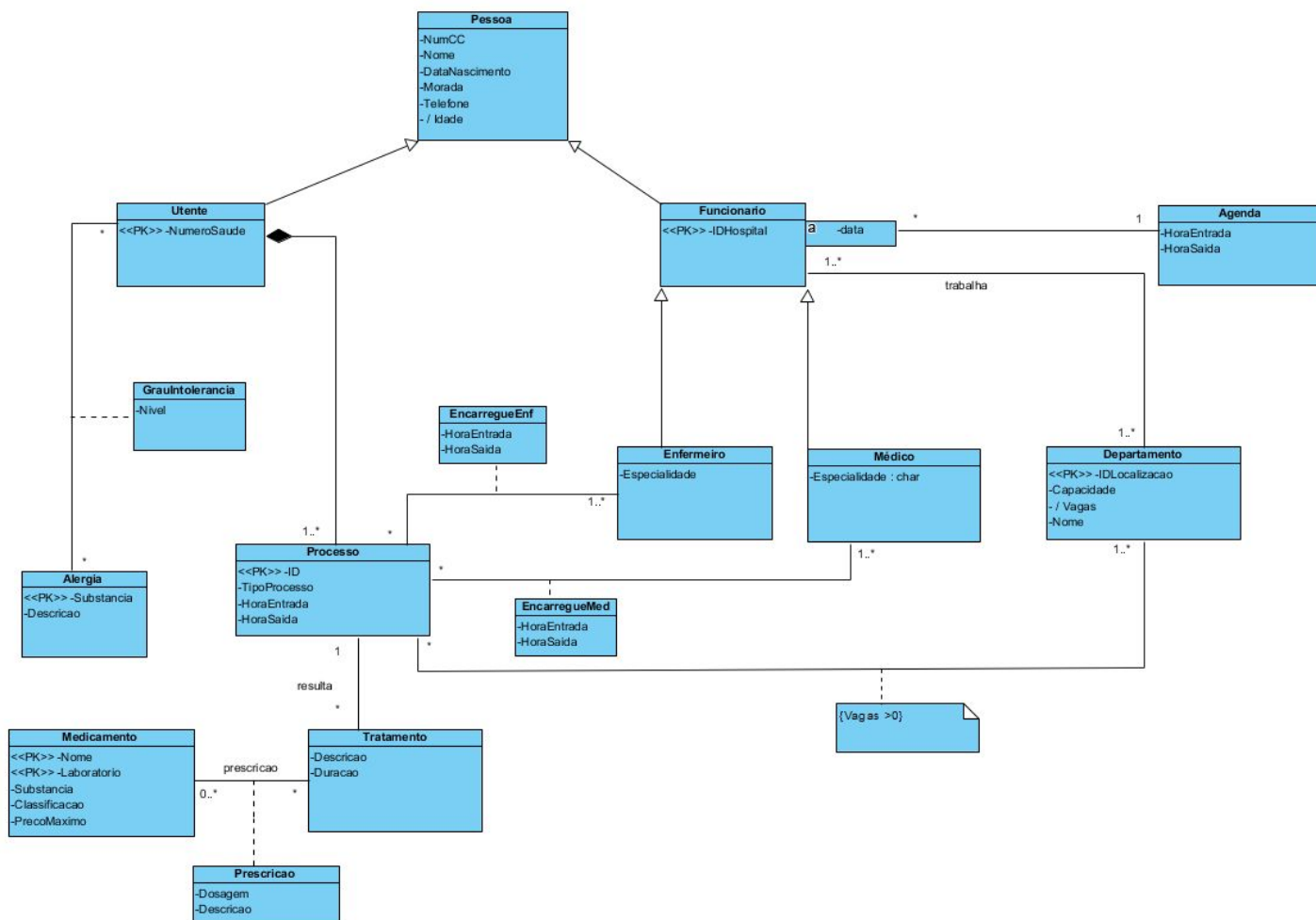
No que toca aos utentes, identificados pelo número do cartão de saúde, são guardadas as várias **alergias** que podem possuir. Cada alergia é caracterizada pela substância em causa e uma pequena descrição do seu efeito. A associação entre o utente e cada uma das suas alergias possui um **grau de intolerância**.

A informação de um utente é composta por vários **processos**, identificados através de um número de processo único. Se for apagada da base de dados a informação relativa a um utente, todos os seus processos serão também eliminados. Cada processo está sempre associado a um departamento, sendo limitado à existência vagas.

A cada processo estão associados um ou mais médicos e um ou mais enfermeiros, e estes dois últimos também poderão ter associados vários processos. É de referir que os funcionários encarregues de um processo podem ser substituídos no decorrer do mesmo. Assim, a atribuição de cada um destes funcionários ao processo tem associada uma hora de entrada e saída no mesmo.

Por último, de cada processo podem resultar vários **tratamentos** (e.g. análises, ecografia, terapias, etc.). Como resultado do processo, podem também ser prescritos **medicamentos**. Estes são identificados por um nome e respectivo laboratório sendo que este par de dados é único para cada medicamento. A **prescrição** guarda informação sobre dosagem e descrição do modo de administração de cada medicamento.

# Diagrama UML



## Considerações

Existem vários aspetos da base de dados que poderiam ser mais restritivos, como por exemplo, não ser possível receitar um medicamento composto por uma substância a que um utente tenha alergia, ou ainda garantir que o funcionário está dentro do seu horário quando é atribuído a um processo. Porém, não consideramos relevante neste contexto que este tipo de limitações seja aplicado ao nível da base de dados. Deste modo, consideramos que o seu objetivo deve ser simplesmente guardar a informação, e permitir que essas decisões sejam praticadas pelo utilizador final. Assim, não é a base de dados que faz a gestão dos horários, mas contém toda a informação para que tal seja possível, também

não é ela que impede o médico de prescrever um medicamento ao qual o utente é alérgico, pois esta é uma decisão médica.

Com isto em conta, tentámos modelar da melhor maneira os dados necessários para o funcionamento de um hospital tendo sido necessário aplicar simplificações em relação a um modelo real (por exemplo, o material médico específico de cada departamento, chefes de departamento e unidades de gestão - tesouraria, administração, admissão de doentes, etc.).

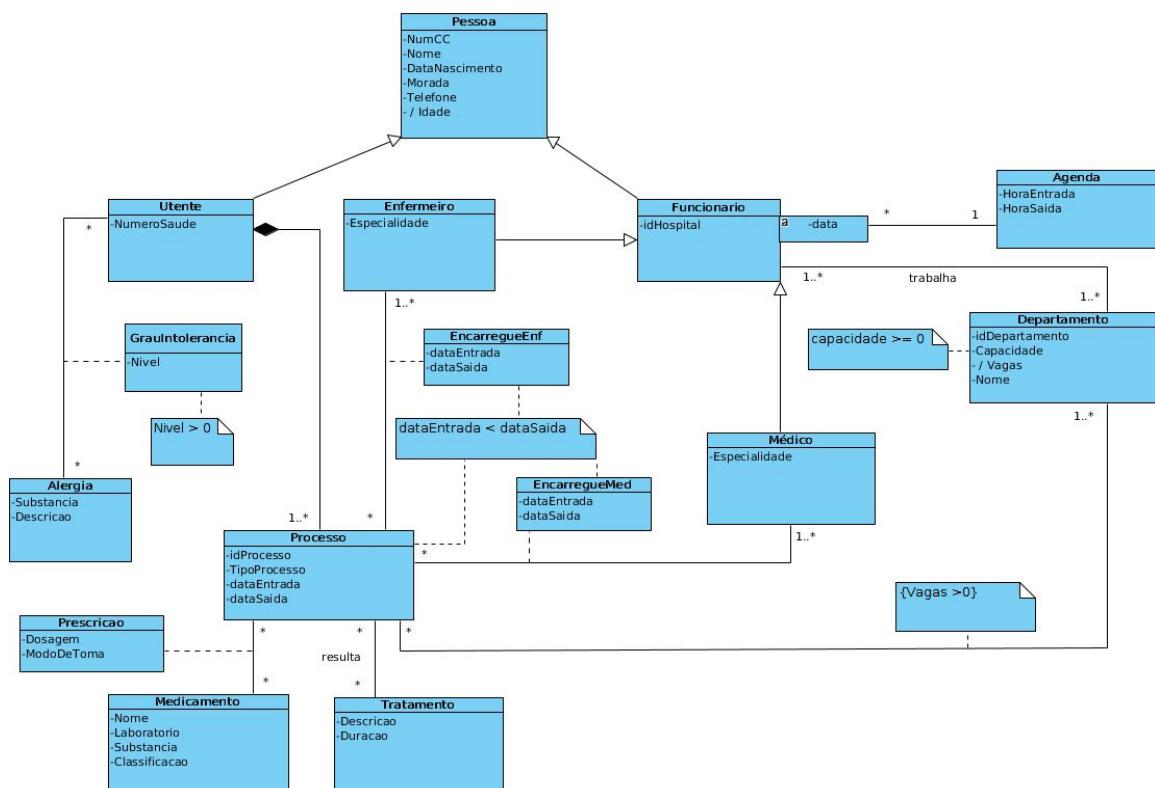
Ainda assim, o modelo que criamos acima contém uma grande variedade de elementos e complexidade a nível conceptual, tornando-se por isso, desafiante de implementar.

Após a revisão do primeiro diagrama UML, as seguintes alterações foram efetuadas:

- A associação entre *Medicamento* e *Tratamento* foi eliminada, passando a existir entre *Medicamento* e *Processo*, mantendo-se a classe associação *Prescrição*. Assim, podem ser prescritos medicamentos a um processo sem que resulte um tratamento (e.g. ecografia, raio-x, etc.);
- A associação entre *Tratamento* e *Processo* passou a ter multiplicidade  *muitos (\*)* do lado do *Processo*. Desta forma, se de *n* processos resultarem o mesmo tratamento, este último não tem de ser registado *n* vezes na base de dados;
- Foram acrescentadas mais três regras de negócio - uma referente a *GraulIntolerancia*, a outra a *Processo*, *EncarregueEnf* e *EncarregueMed* e a terceira a *Departamento*;
- Os atributos *horaSaida* e *horaEntrada* em *Processo*, *EncarregueEnf* e *EncarregueMed* passaram a ser *dataSaida* e *dataEntrada* de forma a representar de forma mais completa a data a que estão associados.

Por forma a melhorar a legibilidade do diagrama e a simplificá-lo, decidimos ainda fazer quatro pequenas alterações:

- O atributo *Descrição* da classe *Prescrição* alterou-se para *ModoDeToma*;
- O atributo *idLocalizacao* de *Departamento* passou a *idDepartamento*;
- O atributo *PrecoMaximo* da classe *Medicamento* foi eliminado;
- Removeram-se do diagrama todas as indicações <<PK>> dos atributos.





# Modelo Relacional

De acordo com o diagrama UML revisto, definimos o seguinte modelo relacional:

Utente(nSaude, nCC, nome, dataNascimento, morada, telefone, /idade)

Funcionario(idHospital, nCC, nome, dataNascimento, morada, telefone, /idade)

Enfermeiro(idHospital->Funcionario, especialidade)

Medico(idHospital->Funcionario, especialidade)

Trabalha(idDepartamento->Departamento, idHospital->Funcionario)

Alergia(substancia, descricao)

Graulntolerancia(nSaude->Utente, substancia->Alergia, nivel)

Processo(idProcesso, tipoProcesso, horaEntrada, horaSaida, nSaude->Utente)

Departamento(idDepartamento, capacidade, nome, /vagas)

EncarregueEnf(idHospital->Enfermeiro, idProc->Processo, horaEntrada, horaSaida)

EncarregueMed(idProc->Processo, idHospital->Medico, horaEntrada, horaSaida)

ProcessoDepartamento(idProc->Processo, idHospital->Funcionario)

Tratamento(idTratamento, descricao, duracao)

Resulta(idTratamento->Tratamento, idProc->Processo)

Medicamento(idMedicamento, nome, laboratorio, substancia, classificacao)

Prescricao(idMedicamento->Medicamento, idProcesso->Processo, dosagem, modoToma)

Data(idHospital->Funcionario, data, idAgenda->Agenda)

Agenda(idAgenda, horaEntrada, horaSaida)

# Dependências funcionais

Do modelo relacional acima retiram-se as seguintes dependências funcionais:

Utente {nCC -> nome, dataNascimento, morada, telefone, idade, nSaude;  
nSaude -> nome, dataNascimento, morada, telefone, idade, nCC;  
dataNascimento -> idade}

Funcionario {nCC -> nome, dataNascimento, morada, telefone, idade, idHospital;  
idHospital -> nome, dataNascimento, morada, telefone, idade, nCC;  
dataNascimento -> idade}

Enfermeiro {idHospital -> especialidade}

Médico {idHospital -> especialidade}

Alergia {substancia -> descricao}

Graulntolerancia {nSaude, substancia -> nivelIntolerancia}

Processo(idProcesso -> tipoProcesso, horaEntrada, horaSaida, nSaude)

Departamento {idDepartamento -> capacidade, nome, vagas}

EncarregueEnf {idHospital, idProc-> horaEntrada, horaSaida}

EncarregueMed {idHospital, idProc-> horaEntrada, horaSaida}

Tratamento {idTratamento -> descricao, duracao}

Medicamento {nome, laboratorio -> substancia; substancia -> classificacao}

Prescricao {idMedicamento, idProc -> dosagem, modoToma}

Data {idHospital,data -> idAgenda}

Agenda {idAgenda -> horaEntrada, horaSaida}

## Formas Normais

Analisando as dependências funcionais de cada relação, chegamos à conclusão que apenas três destas violam a Forma Normal de Boyce-Codd (BCNF) e a 3ª Forma Normal (3NF) - *Utente*, *Funcionário* e *Medicamento*.

As restantes relações, por possuírem poucos atributos e terem poucas dependências entre os mesmos, não violam nenhuma forma normal, estando condição ideal para serem implementadas.

Deste modo, torna-se evidente que é necessário corrigir as relações que violam Formas Normais para evitar anomalias na implementação da base de dados.

Começando pelas relações *Utente* e *Funcionário*, são admitidas como idênticas, uma vez que apenas divergem na existência de um atributo *idHospital* em *Funcionario* e *nSaude* em *Utente* e estes apresentam características semelhantes em cada classe, estando associados às mesmas dependências funcionais. Assim, ambas as classes podem ser tratadas da mesma forma.

Assim sendo, atentando nas dependências funcionais verifica-se que 'dataNascimento->idade' é uma dependência transitiva (um atributo não primo depende funcionalmente de outro atributo não primo) pelo que viola a 3NF. Além disso, como 'dataNascimento' não é superchave da relação, esta também viola a BCNF. As duas restantes dependências têm como lado esquerdo uma chave de cada relação, pelo que não violam nenhuma Forma Normal.

Para corrigir o problema pode-se aplicar uma Decomposição BCNF, da qual obtemos:

*Utente*(*nSaude*, *nCC*, *nome*, *dataNascimento* -> *Nascimento*, *morada*, *telefone*)

*Funcionario*(*idHospital*, *nCC*, *nome*, *dataNascimento* -> *Nascimento*, *morada*, *telefone*)

*Nascimento*(*dataNascimento*, *idade*)

É de notar que a dependência funcional 'dataNascimento->idade' passou apenas a ser relevante na relação *Nascimento*. Visto que o atributo *dataNascimento* é chave desta relação, deixaram de haver violações à 3NF e à BCNF.

Contudo, para melhorarmos a organização da informação, consideramos melhor decompor as relações *Utente* e *Funcionário* usando a dependência funcional '*nCC* -> *nome*, *dataNascimento*, *morada*, *telefone*, *idade*,

nSaude/idHospital' que, no fundo, é a identificação de uma pessoa. Desta forma, ficamos com as relações:

Pessoa(nCC, nome, dataNascimento -> Nascimento, morada, telefone)

Utente(nSaude, nCC)

Funcionario(idHospital, nCC)

Nascimento(dataNascimento, idade)

Passando agora à relação *Medicamento*, a dependência funcional 'nome, laboratorio -> substancia' identifica uma chave composta da relação, pelo que não viola nenhuma das Formas Normais. No entanto, como *substancia* não é superchave da relação, a dependência 'substancia -> classificacao' viola a BCNF e mais uma vez estamos perante uma dependência transitiva, pelo que também viola a 3NF. Assim sendo, voltando a aplicar uma Decomposição BCNF as violações são corrigidas, resultando as relações:

Medicamento(idMedicamento, nome, laboratorio, substancia -> Farmaco)

Farmaco(nome, classificacao)

Agora, a dependência funcional que violava as Formas Normais mantém-se apenas na relação *Fármaco*, na qual identifica uma chave - deixa de violar a BCNF e a 3NF.

Por fim, é de notar que em todas as relações resultantes - *Pessoa*, *Funcionário*, *Utente*, *Nascimento*, *Medicamento* e *Fármaco* - houve preservação das dependências funcionais, pelo que não é posta em causa a integridade do modelo inicial.

# Modelo Normalizado

Por fim, da análise às Formas Normais resulta o modelo relacional normalizado:

Pessoa(nCC, nome, dataNascimento -> Nascimento, morada, telefone)

Utente(nSaude, nCC->Pessoa)

Funcionario(idHospital, nCC->Pessoa)

Nascimento(dataNascimento, idade)

Enfermeiro(idHospital->Funcionario, especialidade)

Medico(idHospital->Funcionario, especialidade)

Trabalha(idDepartamento->Departamento, idHospital->Funcionario)

Alergia(substancia, descricao)

Graulntolerancia(nSaude->Utente, substancia->Alergia, nivel)

Processo(idProcesso, tipoProcesso, horaEntrada, horaSaida, nSaude->Utente)

Departamento(idDepartamento, capacidade, nome, /vagas)

EncarregueEnf(idHospital->Funcionario, idProcesso->Processo, horaEntrada, horaSaida)

EncarregueMed(idProcesso->Processo, idHospital->Funcionario, horaEntrada, horaSaida)

ProcessoDepartamento(idProcesso->Processo, idDepartamento->Departamento)

Tratamento(idTratamento, descricao, duracao)

Resulta(idTratamento->Tratamento, idProc->Processo)

Medicamento(idMedicamento, nome, laboratorio, substancia -> Farmaco)

Farmaco(nome, classificacao)

Prescricao(idMedicamento->Medicamento, idProcesso->Processo, dosagem, modoToma)

Data(idHospital->Funcionario, data, idAgenda->Agenda)

Agenda(idAgenda, horaEntrada, horaSaida)

# Implementação da base de dados

## Restrições

Na implementação da base de dados foram consideradas várias restrições necessárias à manutenção da integridade dos dados armazenados. Todas essas restrições estão presentes na seguinte tabela:

Relação	Restrição	Implementação
Pessoa	As pessoas são identificadas pelo número de cartão de cidadão (atributo <i>nCC</i> )	PRIMARY KEY
	Não existem duas pessoas com o mesmo número de telefone registado	UNIQUE
	Todas as pessoas têm nome	NOT NULL
	Cada pessoa está relacionada com um <i>Nascimento</i>	FOREIGN KEY
Nascimento	A data de nascimento determina a <i>idade</i> e, portanto, toda a relação	PRIMARY KEY
	A <i>idade</i> é sempre maior ou igual a 0	CHECK
Utente	Os utentes são identificados pelo número de saúde	PRIMARY KEY
	O número de cartão de cidadão permite obter informação pessoal do utente	FOREIGN KEY
Funcionário	Os funcionários têm identificadores únicos dentro do hospital	PRIMARY KEY
	O número de cartão de cidadão permite obter informação pessoal do funcionário	FOREIGN KEY
Médico	Cada médico tem um identificador de hospital único, do qual se obtém a informação do funcionário	PRIMARY KEY / FOREIGN KEY

	Todos os médicos têm especialidade	NOT NULL
Enfermeiro	Cada enfermeiro tem um identificador de hospital único, do qual se obtém a informação do funcionário	PRIMARY KEY / FOREIGN KEY
Agenda	Cada agenda tem um ID único	PRIMARY KEY
	As horas de entrada e de saída têm sempre de ser registadas	NOT NULL
Data	Para um funcionário numa determinada data existe apenas uma agenda	PRIMARY KEY
	O identificador do hospital refere-se a um funcionário e o identificador da agenda à informação nela contida	FOREIGN KEY
GrauIntolerancia	O grau de intolerância é identificado pelo número de saúde do utente e a substância da alergia	PRIMARY KEY / FOREIGN KEY
	O nível de intolerância toma valores entre 0 e 5	CHECK
Alergia	Cada alergia é causada por uma única substância	PRIMARY KEY
Departamento	Cada departamento tem um identificador único	PRIMARY KEY
	Todos os departamentos têm obrigatoriamente capacidade e nome	NOT NULL
	A capacidade é sempre superior a 0	CHECK
Processo	Cada processo tem um ID único	PRIMARY KEY
	Um processo tem obrigatoriamente uma data de entrada	NOT NULL
	A data de entrada é sempre anterior à da data de saída	CHECK
EncarregueEnf	Para um dado processo e um dado enfermeiro existe uma só	PRIMARY KEY / FOREIGN KEY

	associação <i>encarregueEnf</i>	
	Há sempre uma data de entrada no processo	NOT NULL
	A data de entrada é sempre anterior à data de saída	CHECK
EncarregueMed	Para um dado processo e um dado médico existe uma só associação <i>encarregueMed</i>	PRIMARY KEY / FOREIGN KEY
	Há sempre uma data de entrada no processo	NOT NULL
	A data de entrada é sempre anterior à data de saída	CHECK
Processo Departamento	Existe uma só entrada de um processo num departamento	PRIMARY KEY / FOREIGN KEY
Trabalha	Existe apenas uma relação de trabalho entre um dado funcionário e um departamento	PRIMARY KEY / FOREIGN KEY
Tratamento	Cada tratamento tem um ID único	PRIMARY KEY
	A duração é sempre superior a 0	CHECK
Resulta	Entre um dado processo e um tratamento há apenas uma associação	PRIMARY KEY / FOREIGN KEY
Medicamento	Cada medicamento tem um ID único	PRIMARY KEY
	A substância refere-se à informação do fármaco do medicamento	FOREIGN KEY
	No mesmo laboratório não são fabricados dois medicamentos com o mesmo nome	UNIQUE
Fármaco	Cada fármaco é identificado pelo seu nome	PRIMARY KEY
Prescrição	Existe apenas uma prescrição para um determinado processo e medicamento	PRIMARY KEY / FOREIGN KEY



# Interrogações

Por forma a demonstrar a utilidade da nossa base de dados, apresentamos as seguintes interrogações:

1. Qual a capacidade total de camas no hospital?
2. Quais as alergias de cada utente?
3. Quais os funcionários envolvidos num determinado processo?
4. A duração média dos processos finalizados?
5. Qual o número de processos por funcionário, ordenados por ordem decrescente?
6. Quais os medicamentos que cada utente internado (cujo processo ainda se encontra ativo) toma?
7. Qual o processo, de entre os terminados, mais longo de cada departamento?
8. Qual o número de ocorrências de cada tratamento entre '2019-01-01' e '2019-04-15'?
9. Quais os médicos disponíveis (i.e. que não se encontram encarregues de nenhum processo) no dia '2019-04-15' a partir das '16:00h'?
10. Quais os enfermeiros que trabalharam todos os dias da semana de '2019-04-15' a '2019-04-21'?

Todas as datas e horas específicas de cada interrogação são generalizáveis pelo software que utilizar a base de dados, bastando apenas trocar os seus valores para obter resultados diferentes.

# Triggers

Foram criados três triggers que consideramos pertinentes para a base de dados em questão.

## 1º Trigger

O hospital tem de conseguir ter acesso a todos os registos passados, mesmo quando se tratam de registos associados a funcionários que já não se encontram no hospital. Isto implica que os funcionários não possam ser eliminados da base de dados para que não ocorra perda de informação, mas manter a possibilidade de em termos práticos remover membros do staff do hospital.

Deste modo, é necessário criar um trigger para que quando um funcionário é eliminado, apenas atualiza um atributo - *ativo* - que permite identificar que esse funcionário já não faz parte do hospital (trigger *RemoveFunc*).

Com isto, surge a necessidade de criar outros triggers decorrentes de restrições do último, pois não faz sentido permitir que seja eliminado um elemento que já não se encontra ativo (trigger *RestraintRem*) , assim como inserir num processo um funcionário que não se encontre ativo (triggers *RestraintAddProcMed* e *RestraintAddProcEnf*).

## 2º Trigger

De cada departamento são registadas as vagas para facilitar a consulta dessa informação. Contudo, as vagas derivam da capacidade do departamento e do número de processos que lhe foram associados.

Assim sendo, sempre que se associa um processo a um departamento (tabela *ProcessoDepartamento*), as vagas nesse departamento são decrementadas em 1 unidade (trigger *DecrementVagas*). Por outro lado, sempre que um processo deixa de estar associado a um departamento, as vagas nesse departamento são incrementadas em 1 unidade (trigger *IncrementVagas*).

Por fim, visto que um departamento não pode ter vagas negativas, sempre que há uma alteração ao número de vagas é verificado se o seu novo valor é válido, isto é, se é superior ou igual a 0 e se é inferior ou igual à capacidade do departamento. Caso o valor não seja válido é lançado um erro e a operação é desfeita.

### 3ºTrigger

Quando se altera a capacidade de um dado departamento, o novo valor para esta propriedade não pode ser inferior ao número de quartos ocupados atualmente. Caso isso aconteça, é lançado um erro e o valor de capacidade não é alterado (trigger *check\_capacity*).

## Conclusão

Com trabalho podemos desenvolver uma base de dados, permitindo aprofundar o nosso conhecimento dos vários temas lecionados ao longo do semestre. Consideramos que divisão deste trabalho em três fases é bastante pertinente, uma vez que facilita o processo da criação da base de dados e permite melhor assimilar os conhecimentos aplicados em cada fase. Como consequência, o aproveitamento desta unidade curricular é significativamente melhor.

Relativamente às dificuldades sentidas, é de salientar a conversão do modelo UML para o modelo relacional. De facto, sentimos algumas dificuldades em tomar as decisões mais acertadas nalgumas situações que surgiram nesta conversão, sendo que alguns dos problemas que nos surgiram mais à frente tiveram origem nesta mesma etapa.

Por último, graças a um esforço coletivo e uma boa divisão de trabalho, estamos muito satisfeitos com a base de dados final e com as interrogações e triggers que dela fazem uso.