**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

Mestrado Integrado em Engenharia Informática e Computação – 2º ano

**Hospital HPL**

**Definição do Modelo Conceptual**

**Turma 7, Grupo 705:**

Henrique Costa Sousa - 201906681

Leandro Martins de Oliveira - 200903027

Pedro Jorge Ribeiro Botelho de Moniz Pacheco - 201806824

**Unidade Curricular:** Base de Dados

**Docentes:**

Carla Alexandra Teixeira Lopes

Gonçalo Filipe Loureiro Campos Gonçalves

**Porto**

**2020/2021**

# **Índice**

[Índice 2](#_Toc68467087)

[Contexto 3](#_Toc68467088)

[Diagrama UML 5](#_Toc68467089)

[Esquema Relacional 6](#_Toc68467090)

[Análise de Dependências Funcionais e Formas Normais 9](#_Toc68467091)

[Restrições 10](#_Toc68467092)

**Interrogações**

**Gatilhos**

# **Contexto**

Pretende-se armazenar dados relativo a um hospital cujo nome fictício será Hospital HPL.

A base de dados deve ser capaz de lidar com vários tipos de pessoas, sendo as mesmas inseridas em duas categorias: trabalhadores e clientes. Cada pessoa da base de dados tem um número de identificação, nome, data de nascimento e uma data de morte (caso se aplique), idade, género, número de telefone, NIF, e-mail, morada e número de seguro.

Os trabalhadores do hospital deverão ter um salário (exceto os voluntários) e um certo número de turnos. Um turno é caracterizado pelo dia e hora de entrada e de saída e terá menos de 48 horas. Os trabalhadores poderão ser gerentes, pessoal da manutenção, voluntários (associados a uma instituição de voluntariado), enfermeiros e médicos (com uma dada especialização). Todos deverão estar ligados a um dado departamento dentro do hospital. Um departamento encontra-se num determinado piso e tem um determinado gerente.

Os clientes do hospital podem ser pacientes ou visitantes. Quando alguém visita um paciente deverá ser registado o início e o fim da visita. Um paciente só pode ter no máximo três visitantes únicos e um visitante só pode visitar um único paciente. Para cada paciente deverá ser guardado os serviços que utilizou.

Um serviço terá um certo diagnóstico e medicação, preço, datas de entrada e de saída associados. Um serviço poderá ser um de três tipos: consulta, cirurgia, serviço de ambulância. Um serviço de ambulância contém um número de identificação associado à ambulância e a sua prioridade. Conforme o nível de urgência da necessidade de uma ambulância, existe uma prioridade de 1 até 5, onde 1 é pouco prioridade e 5 muita prioridade. Interessa também guardar as informações sobre a manutenção da ambulância. Uma ambulância apenas pode receber manutenção de 3 membros do pessoal da manutenção. Para cada par que consiste de um membro da manutenção e da ambulância deve ser guardada a data e se foi realizada desinfeção ou reabastecimento. Por outro lado, um serviço pode ter associado um dado número de médicos e/ou enfermeiros. E deverá ser mantido o

local em que decorreu o serviço. O local poderá ser externo (tendo assim que se registar a respetiva morada), uma sala de cirurgia, para o qual interessa saber a especialização; uma unidade de cuidados intensivos, com possível utilização de oxigénio ou de terapia intravenosa; unidade de cuidados normais ou ainda um escritório.

# **Diagrama** **UML**

# 

# **Esquema Relacional**

**Person** (id, name, birth\_date, death\_date, age, gender, phone\_number, NIF, email, address, insurance\_id)

id -> name, birth\_date, death\_date, age, gender, phone\_number, NIF, email, address, insurance\_id

**Department** (id, floor)

id -> floor

**Location** (id, room\_number, bed\_number)

id -> room\_number, bed\_number

**Shift** (id, day\_of\_the\_week\_in, time\_in, day\_of\_the\_week\_out, time\_out)

id -> day\_of\_the\_week\_in, time\_in, day\_of\_the\_week\_out, time\_out

**Service** (id, diagnosis, medication, price, date\_in, date\_out, location -> Location patient -> Patient)

id -> diagnosis, medication, price, date\_in, date\_out, location, patient

**Specialization** (speciality)

speciality -> speciality

**Worker** (person->Person)

person -> person

**WorkerShift** (worker->Worker, shift->Shift)

worker, shift -> worker, shift

**Manager** (person->Worker, salary, department->Department)

person -> department

**Maintenance** (person->Worker, salary, department->Department)

person -> department

**Volunteer** (person->Worker, association\_name, department->Department)

person -> association\_name, department

**Nurse** (person->Worker, salary, department->Department)

person -> salary, department

**NurseService** (nurse->Nurse, service->Service)

nurse, service -> nurse, service

**Doctor** (person->Worker, salary, department->Department)

person-> salary, department

**DoctorService** (doctor->Doctor, service->Service)

doctor, service -> doctor, service

**DoctorSpecialization** (doctor->Doctor, specialization->Specialization)

doctor, specialization -> doctor, specialization

**Client** (person->Person)

person -> person

**Patient** (person->Client)

person -> person

**Visitor** (person->Client)

person -> person

**VisitTime** (patient->Patient, visitor->Visitor, start\_visit\_date, end\_visit\_date, order)

patient, visitor -> start\_visit\_date, end\_visit\_date, order

**Appointment** (service->Service)

service -> service

**Surgery** (service->Service)

service -> service

**Ambulance** (service->Service, amb\_id, priority)

service -> amb\_id, priority

**MaintenanceJob** (ambulance -> Ambulance, maintenance -> Maintenance, date, did\_disinfection, did\_restock)

ambulance, maintenance -> date, did\_disinfection, did\_restock

**Extern** (location->Location, address)

location -> address

**SurgeryRoom** (location->Location, specialization->Specialization)

location->specialization

**IntensiveCareRoom** (location->Location, o2, iv)

location -> o2, iv

**NormalCareRoom** (location->Location)

location -> location

**Office** (location->Location)

location -> location

# **Análise de Dependências Funcionais e Formas Normais**

Para as relações Specialization, Worker, WorkerShift, NurseService, DoctorService, Client, Patient, Visitor, Appointment, Surgery, NormalCareRoom e Office as dependências funcionais estão na BCNF e na 3ª forma normal, uma vez que se trata de dependências funcionais triviais.

Para as restantes relações, uma vez que o lado esquerdo da relação funcional é também a sua chave, é possível concluir que se encontra na forma BCNF e na 3ª forma normal.

Deste modo, não foi preciso decompor nenhuma das relações.

# **Restrições**

|  |
| --- |
|  |

**Person**

id PRIMARY KEY  
id é a chave primária

Name NOT NULL  
O nome não pode ser nulo

age INTEGER CHECK(age > 0)  
A idade não pode ser negativa

gender TEXT CHECK(gender == 'M' OR gender == 'F' OR gender == ‘O’)  
Não pode haver uma pessoa com dois géneros ao mesmo tempo, e o seu género será masculino, feminino, ou outro

NIF UNIQUE  
Não pode existir duas pessoas com o mesmo NIF

NIF VARCHAR(9) CHECK(length(nif)==9)  
NIF tem que ter 9 caracteres

email TEXT LIKE “%@%.%”  
O email tem que conter o símbolo @ e um ponto (“.”) após este

insurance\_id INTEGER UNIQUE  
O número de seguro tem que ser único

**Department**

id PRIMARY KEY  
O id é a chave primária

**Location**

id PRIMARY KEY  
O id é a chave primária

room\_number CHECK(room\_number > 0)  
Os números dos quartos têm de ser maiores que zero

bed\_number CHECK(bed\_number > 0)  
Os números das camas têm de ser mairoes que zero

bed\_number UNIQUE  
Os números das camas têm de ser únicos

**Shift**

Id PRIMARY KEY  
O id é a chave primária

CONSTRAINT dayInIsADay CHECK(day\_of\_the\_week\_in >= 0 AND day\_of\_the\_week\_in <= 6)  
day\_of\_the\_week\_in é um dia compreendido entre 0 e 6

CONSTRAINT dayOutIsADay CHECK(day\_of\_the\_week\_out >= 0 AND day\_of\_the\_week\_out <= 6)  
day\_of\_the\_week\_out é um dia compreendido entre 0 e 6

CONSTRAINT shiftLessThan48 CHECK((((day\_of\_the\_week\_out - day\_of\_the\_week\_in) < 2 AND (day\_of\_the\_week\_out - day\_of\_the\_week\_in) >= 0) OR ((day\_of\_the\_week\_out - day\_of\_the\_week\_in) + 7 < 2))OR(((day\_of\_the\_week\_out - day\_of\_the\_week\_in) == 2 OR (day\_of\_the\_week\_out - day\_of\_the\_week\_in) + 7 == 2) AND strftime('%s', time\_in) >= strftime('%s', time\_out)))  
O turno tem menos de 48 horas

**Specialization**

speciality PRIMARY KEY  
A especialização é a chave primária

**Worker**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES PERSON  
A person é uma referência a um tuplo da tabela Person

**WorkerShift**

worker REFERENCES Worker  
O worker é uma referência a um tuplo da tabela Worker

shift REFERENCES Shift  
O shift é uma referência a um tuplo da tabela Shift

PRIMARY KEY (worker, shift)  
A chave primária é um par worker, shift

**Manager**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES Worker  
A person é uma referência a um tuplo da tabela Worker

department NOT NULL  
O manager tem de gerir um certo departamento

department REFERENCES Department  
O department é um tuplo da tabela Department

CONSTRAINT needsMoney CHECK(salary >= 0)  
O salário tem que ser positivo

**Maintenance**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES Worker  
A person é uma referência a um tuplo da tabela Worker

department NOT NULL  
Um funcionário da manutenção tem de pertencer a um dado departamento

department REFERENCES Department  
O department é um tuplo da tabela Department

CONSTRAINT needsMoney CHECK(salary >= 0)  
O salário tem que ser positivo

**Volunteer**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES Worker  
A person é uma referência a um tuplo da tabela Worker

department NOT NULL  
Um voluntário tem de pertencer a um dado departamento

department REFERENCES Department  
O department é um tuplo da tabela Department

**Nurse**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES Worker  
A person é uma referência a um tuplo da tabela Worker

department NOT NULL  
Um enfermeiro tem de pertencer a um dado departamento

department REFERENCES Department  
O department é um tuplo da tabela Department

CONSTRAINT needsMoney CHECK(salary >= 0)  
O salário tem que ser positivo

**Doctor**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES Worker  
A person é uma referência a um tuplo da tabela Worker

department NOT NULL  
Um médico tem que pertencer a um dado departamento

department REFERENCES Department  
O department é um tuplo da tabela Department

CONSTRAINT needsMoney CHECK(salary >= 0)  
O salário tem que ser positivo

**DoctorSpecialization**

doctor REFERENCES Person  
O doctor é uma referência a um tuplo da tabela Doctor

specialization REFERENCES Specialization  
A specialization é uma referência a um tuplo da tabela Specialization

PRIMARY KEY (doctor, specialization)  
A chave primária é um par doctor, specialization

**Client**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES Person  
A person é uma referência a um tuplo da tabela Person

**Patient**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES Person  
A person é uma referência a um tuplo da tabela Person

**Visitor**

person PRIMARY KEY  
A person é a chave primária

person REFERENCES Person  
A person é uma referência a um tuplo da tabela Person

**VisitTime**

patient REFERENCES Patient  
A person é uma referência a um tuplo da tabela Person

person REFERENCES Visitor  
A person é uma referência a um tuplo da tabela Person

PRIMARY KEY (patient, visitor)  
A chave primária é um par patient, visitor

CONSTRAINT notSamePerson CHECK(patient IS NOT visitor)  
O paciente e o visitante não sou a mesma pessoa

CONSTRAINT validVisitTime CHECK(end\_visit\_date > start\_visit\_date)  
O final da visita tem de ser após o início da mesma

**Service**

id PRIMARY KEY  
O id é a chave primária

price CHECK (price >= 0)  
O preço de um serviço tem que ser não negativo

date\_in NOT NULL  
A admissão num dado serviço não pode ser nula

location REFERENCES Location  
A location é uma referência a um tuplo da tabela Location

patient REFERENCES Patient  
O patient é uma referência a um tuplo da tabela Patient

CONSTRAINT validServiceTime CHECK(date\_out > date\_in OR date\_out IS NULL)  
A data de saída é maior que a data de entrada; se o paciente não tiver saído ainda, a data de saída é nula

**NurseService**

nurse REFERENCES Nurse  
O nurse é uma referência a um tuplo da tabela Nurse

service REFERENCES Service  
O service é uma referência a um tuplo da tabela Service

PRIMARY KEY (nurse, service)  
A chave primária é um par nurse, service

**DoctorService**

doctor REFERENCES Doctor  
O doctor é uma referência a um tuplo da tabela Doctor

service REFERENCES Service  
O service é uma referência a um tuplo da tabela Service

PRIMARY KEY (doctor, service)  
A chave primária é um par doctor, service

**Appointment**

service PRIMARY KEY  
O service é a chave primária

service REFERENCES Service  
O service é uma referência a um tuplo da tabela Service

**Surgery**

service PRIMARY KEY  
O service é a chave primária

service REFERENCES Service  
O service é uma referência a um tuplo da tabela Service

**Ambulance**

service PRIMARY KEY  
O service é a chave primária

service REFERENCES Service  
O service é uma referência a um tuplo da tabela Service

amb\_id CHECK(amb\_id > 0)  
O id de uma ambulância é positivo

amb\_id UNIQUE  
O id de uma ambulância é único

priority CHECK(priority >=1 AND priority <= 5)  
A prioridade de uma ambulância está compreendida entre 1 e 5, onde 1 é pouco prioridade e 5 muita prioridade

**MaintenanceJob**

ambulance REFERENCES Ambulance  
A ambulance é uma referência a um tuplo da tabela Ambulance

maintenance REFERENCES Maintenance  
O maintenance é uma referência a um tuplo da tabela Maintenance

PRIMARY KEY (ambulance, maintenance)  
A chave primária é um par ambulance, maintenance

CONSTRAINT disinfectionBoolean CHECK (did\_disinfection == 0 OR did\_disinfection == 1)  
A desinfeção da ambulância ocorreu ou não

CONSTRAINT restockBoolean CHECK (did\_restock == 0 OR did\_restock == 1)  
O reabastecimento da ambulância ocorreu ou não

**Extern**

location PRIMARY KEY  
A location é a chave primária

location REFERENCES Location  
A location é uma referência a um tuplo da tabela Location

address NOT NULL  
Um lugar externo, não no hospital, tem de ter uma morada

**SurgeryRoom**

location PRIMARY KEY  
A location é a chave primária

location REFERENCES Location  
A location é uma referência a um tuplo da tabela Location

specialization NOT NULL  
Uma sala de cirugia tem que ser especializada numa certa área

specialization REFERENCES Specialization  
Uma specialization é uma referência a um tuplo da tabela Specialization

**IntensiveCareRoom**

location PRIMARY KEY  
A location é a chave primária

location REFERENCES Location  
A location é uma referência a um tuplo da tabela Location

CONSTRAINT o2Boolean CHECK (o2 == 0 OR o2 == 1)  
O oxigénio está ligado ou não

CONSTRAINT ivBoolean CHECK (iv == 0 OR iv == 1)  
A admnistração está presente ou não

**NormalCareRoom**

location PRIMARY KEY  
A location é a chave primária

location REFERENCES Location  
A location é uma referência a um tuplo da tabela Location

**Office**

location PRIMARY KEY  
A location é a chave primária

location REFERENCES Location  
A location é uma referência a um tuplo da tabela Location

O atributo age será computado numa View, e por isso, apenas implementado na próxima entrega.

Da mesma forma, existem algumas restrições que serão implementadas através de triggers, que também deixaremos para depois:

* A chave para Specialization é case-insensitive e guardada como tal
* No máximo três visitantes poderão visitar um paciente
* No máximo três membros do pessoal da manutenção realizarão a manutenção de uma ambulância
* As localizações e os trabalhadores deverão ser disjuntos, isto é, uma localização não poderá pertencer a mais do que uma subclasse da mesma; e um trabalhador não poderá pertencer a mais do que uma subclasse do mesmo.

**Interrogações**

1. Quais os trabalhadores do hospital?
2. Qual o número de especializações de cada médico?
3. Quais os trabalhadores ordenados por departamento?
4. Quais os pacientes que tiveram mais que um visitante?
5. Quais os serviços ordenados por preço de forma descendente e com o paciente que usou o serviço?
6. Quais as ambulâncias disponíveis e a suas prioridades, ordenadas de prioridade superior para prioridade inferior?
7. Quais os gerentes e os médicos associados entre si, mas apenas para os médicos que têm consultas diretas?
8. Quais os enfermeiros que participaram em todos os serviços?
9. Quais os trabalhadores relacionados com um paciente?
10. Qual a soma dos salários a ser pago aos trabalhadores que participaram num serviço?

**Gatilhos**

1. Gatilho que verifica que um trabalhador já inscrito não pode inscrever noutro sítio.
2. Gatilho que verifica se o número máximo de pessoal de manutenção por ambulância é, no máximo, 3; e se o número máximo de visitantes por um certo paciente é também, no máximo, 3.
3. Gatilho que verifica se a especialização do médico é a mesma que a especialização da sala de cirurgia.