

2018/19

Gestão de Recursos de Biblioteca

Base de Dados

BRUNO FILIPE CARDOSO MICAELO – UP201706044

MIGUEL DELGADO PINTO – UP201706156

PEDRO MIGUEL RODRIGUES FERRAZ ESTEVES – UP201705160

# Introdução

O projeto assenta na criação de uma base dados para auxiliar a gestão de uma biblioteca.

Como qualquer biblioteca, nela podem ser requisitados livros de diversos géneros. Os utilizadores têm também ao seu dispor vários elementos multimédia, como filmes, álbuns de música e software.

Se não pretenderem fazer uma requisição, podem usufruir de salas disponíveis para reserva, equipadas com computadores, projetores, televisões, entre outros.

# Planeamento da Biblioteca

A informação de qualquer indivíduo relacionado com a biblioteca será armazenada na classe **Pessoa**. Estes são caracterizados pelo nome, data de nascimento, número de telemóvel e número de cartão de cidadão.

Os trabalhadores na biblioteca, classe **Funcionário**, derivada de Pessoa, possuem um ID próprio, um salário, hora de entrada e hora de saída e informação complementar, como número de contribuinte e morada.

A classe **Utilizador**, derivada de Pessoa, facilita a organização do sistema e a identificação dos usuários da biblioteca.

Cada utilizador pode realizar uma **Requisição** de um **Exemplar** de uma certa **Publicação**.

As requisições têm um ID, uma data e uma hora caraterísticas. As publicações possuem um ID, um nome, um género e uma idade mínima. É necessário que o exemplar esteja disponível para requisição, uma vez que certos exemplares devem estar permanentemente na biblioteca. Possuem também dias de atraso e multa, no caso de um utilizador se atrasar na devolução de um exemplar.

As publicações estão organizadas por classes derivadas do tipo Publicação, que podem ser: **Livro**, **Filme**, **Álbum** ou **Software**.

Um livro é caracterizado por uma editora e pelo número da edição, sendo que está associado a pelo menos um **Autor** (cada autor tem um ID e um nome).

Um filme distingue-se pelo realizador e pelo estúdio.

Um álbum define-se pelo seu produtor.

Note-se que um álbum conta com a participação de pelo menos um **Artista**, sendo este caracterizado por um ID e pelo seu nome.

Um software classifica-se pela sua versão e pelo developer.

Os utilizadores podem também usufruir de **Salas** disponíveis para **Reserva**.

As salas são caracterizadas por um número, um tipo de sala e capacidade. Têm também **Equipamento** específico (cada um com um ID, marca e modelo). Para facilitar a organização, a cada equipamento está associado uma classe **TipoEquipamento**. Existe um número limitado de tipos de equipamento: Leitor de CDs, Leitor de DVDs, Leitor de VHS, Computador, Projetor e Televisão, cada um com um propósito específico.

A reserva destas salas tem um ID, um motivo, uma data e hora específicas e uma duração.

Como se trata de uma biblioteca de excelência, os seus bens estão sempre nas melhores condições. Para tal, os funcionários realizam atividades de manutenção (classe **Tipo de Manutenção**) em certos exemplares. São caracterizadas pelo seu nome.

# Diagrama UML

O diagrama UML encontra-se em anexo (no ficheiro **Biblioteca\_UML.jpg**).

# Modelo Relacional

**Pessoa**(CartaoCidadao, Nome, DataNascimento, Telefone)

Dependências funcionais:

* CartaoCidadao -> {Nome, DataNascimento, Telefone}

A chave principal da relação Pessoa é CartaoCidadao.

**Utilizador**(CartaoCidadao -> Pessoa)

A chave principal da relação Utilizador é CartaoCidadao, que serve também como chave estrangeira para a relação Pessoa.

**Funcionario**(CartaoCidadao -> Pessoa, Salario, Contribuinte, Morada, HoraEntrada, HoraSaida)

Dependências funcionais:

* CartaoCidadao -> {Contribuinte, Morada, HotaEntrada, HoraSaida}

Tal como para a relação Utilizador, a chave principal é CartaoCidadao, que é também uma chave estrangeira para Pessoa.

**Reserva**(IDReserva, Motivo, Data, Hora, Duracao, CCutilizador -> Utilizador)

Dependências funcionais:

* IDReserva -> {Motivo, Data, Hora, Duracao, CCutilizador}

A chave principal para Reserva é IDReserva, tendo também CCutilizador como chave estrangeira para uma relação Utilizador.

**Sala**(Numero, Tipo, Capacidade)

Dependências funcionais:

* Numero -> {Tipo, Capacidade}

A chave principal para Sala é Numero.

**ReservaDeSala**(IDReserva -> Reserva, NumeroSala -> Sala)

A relação ReservaDeSala tem como chave principal a chave composta por IDReserva e NumeroSala, chaves estrangeiras para Reserva e Sala, respetivamente.

**Equipamento**(IDEquipamento, Marca, Modelo, NumeroSala -> Sala, NomeTipo -> TipoEquipamento)

Dependências funcionais:

* IDEquipamento -> {Marca, Modelo, NumeroSala, NomeTipo}
* Modelo -> {Marca, NomeTipo}

A chave principal para Equipamento é IDEquipamento, tendo NumeroSala e NomeTipo como chaves estrangeiras para Sala e TipoEquipamento, respetivamente.

**TipoEquipamento**(Nome, Proposito)

Dependências funcionais:

* Nome -> {Proposito}

A chave principal para TipoEquipamento é Nome.

**Requisicao**(IDRequisicao, Data, Hora, DiasAtraso, Multa, CCutilizador -> Utilizador)

Dependências funcionais:

* IDRequisicao -> {Data, Hora, DiasAtraso, Multa, CCutilizador}

A chave principal para Requisicao é IDRequisicao, tendo como chave estrangeira para Utilizador CCutilizador.

**Exemplar**(IDExemplar, PossivelRequisitar, IDSala -> Sala, IDPublicacao -> Publicacao)

Dependências funcionais:

* IDExemplar -> {PossivelRequisitar, IDSala, IDPublicacao}

A chave principal para Exemplar é IDExemplar, com IDSala e IDPublicacao como chaves estrangeiras para as relações Sala e Publicacao, respetivamente.

**RequisicaoDeExemplar**(IDRequisicao -> Requisicao, IDExemplar -> Exemplar)

A chave principal de RequisicaoDeExemplar é a chave composta por IDRequisicao e IDExemplar, chaves estrangeiras para Requisicao e Exemplar, respetivamente.

**TipoDeManutencao**(Nome)

A chave principal para TipoManutencao é Nome.

**AtoDeManutencao**(CCfuncionario -> Funcionario, NomeManutencao -> TipodeManutencao, IDExemplar -> Exemplar)

AtoDeManutencao tem como chave principal a chave composta por Ccfuncionario, NomeManutencao e IDExemplar, que servem também como chaves estrangeiras para as relações Funcionario, TipodeManutencao e Exemplar.

**Publicacao**(IDPublicacao, Nome, Genero, IdadeMinima, Quantidade)

Dependências funcionais:

* IDPublicacao -> {Nome, Genero, IdadeMinima, Quantidade}

A chave principal da relação Publicacao é IDPublicacao.

**Livro**(IDPublicacao -> Publicacao, Editora, Edicao)

Dependências funcionais:

* IDPublicacao -> {Editora, Edicao}

A chave principal de Livro é IDPublicacao, que é também uma chave estrangeira para Publicacao.

**Autor**(IDAutor, Nome)

Dependências funcionais:

* IDAutor -> {Nome}

A chave principal para Autor é IDAutor.

**Autoria**(IDPublicacao -> Livro, IDAutor -> Autor)

Autoria tem como chave principal a chave composta por IDPublicacao e IDAutor, que servem também como chaves estrangeiras para as relações Livro e Autor, respetivamente.

**Software**(IDPublicacao -> Publicacao, Versao, Developer)

Dependências funcionais:

* IDPublicacao -> {Versao, Developer}

A chave principal de Software é IDPublicacao, que é também uma chave estrangeira para Publicacao.

**Album**(IDPublicacao -> Publicacao, Produtor)

Dependências funcionais:

* IDPublicacao -> {Produtor}

A chave principal de Album é IDPublicacao, que é também uma chave estrangeira para Publicacao.

**Artista**(IDArtista, Nome)

Dependências funcionais:

* IDArtista -> {Nome}

A relação Artista tem como chave principal IDArtista.

**Interpreta**(IDPublicacao -> Album, IDArtista -> Artista)

A relação Interpreta tem como chave principal a chave composta por IDPublicacao e IDArtista, também chaves estrangeiras para Album e Artista, respetivamente.

**Filme**(IDPublicacao -> Publicacao, Realizador, Estudio)

Dependências funcionais:

* IDPublicacao -> {Publicacao, Realizador, Estudio}

A chave principal de Filme é IDPublicacao, que é também uma chave estrangeira para Publicacao.

# Análise das Relações

De modo a analisar as dependências funcionais de cada relação e as conseguintes formas normais, é necessário analisar os fechos dos lados esquerdos de cada dependência funcional.

Relação **Pessoa**:

* {CartaoCidadao}+ = {CartaoCidadao, Nome, DataNascimento, Telefone}

Relação **Utilizador**:

* {CartaoCidadao}+ = {CartaoCidadao}

Relação **Funcionario**:

* {CartaoCidadao}+ = {CartaoCidadao, Salario, Contribuinte, Morada, HoraEntrada, HoraSaida}

Relação **Reserva**:

* {IDReserva}+ = {IDReserva, Motivo, Data, Hora, Duracao, CCutilizador}

Relação **Sala**:

* {Numero}+ = {Numero, Tipo, Capacidade}

Relação **ReservaDeSala**:

* {IDReserva, NumeroSala}+ = {IDReserva, NumeroSala}

Relação **Equipamento**:

* {IDEquipamento}+ = {IDEquipamento, Marca, Modelo, NumeroSala, NomeTipo}
* {Modelo}+ = {Modelo, Marca, NomeTipo}

Relação **TipoEquipamento**:

* {Nome}+ = {Nome, Proposito}

Relação **Requisicao**:

* {IDRequisicao}+ = {IDRequisição, Data, Hora, DiasAtraso, Multa, CCutilizador}

Relação **Exemplar**:

* {IDExemplar}+ = {IDExemplar, PossivelRequisitar, IDSala, IDPublicacao}

Relação **RequisicaoDeExemplar**:

* {IDResquisicao, IDExemplar}+ = {IDResquisicao, IDExemplar}

Relação **TipoDeManutencao**:

* {Nome}+ = {Nome}

Relação **AtoDeManutencao**:

* {Ccfuncionario, NomeManutencao, IDExemplar}+ = {Ccfuncionario, NomeManutencao, IDExemplar}

Relação **Publicacao**:

* {IDPublicacao}+ = {IDPublicaçao, Nome, Genero, IdadeMinima, Quantidade}

Relação **Livro**:

* {IDPublicacao}+ = {IDPublicacao, Editora, Edicao}

Relação **Autor**:

* {IDAutor}+ = {IDAutor, Nome}

Relação **Autoria**:

* {IDPublicacao, IDAutor}+ = {IDPublicacao, IDAutor}

Relação **Software**:

* {IDPublicacao}+ = {IDPublicacao, Versao, Developer}

Relação **Album**:

* {IDPublicacao}+ = {IDPublicacao, Produtor}

Relação **Artista**:

* {IDArtista}+ = {IDArtista, Nome}

Relação **Interpreta**:

* {IDPublicacao, IDArtista}+ = {IDPublicacao, IDArtista}

Relação **Filme**:

* {IDPublicacao}+ = {IDPublicacao, Realizador, Estudio}

Tendo em conta a análise dos fechos de cada dependência funcional, pode-se concluir que a maioria destas respeita a Terceira Forma Normal, uma vez que respeitam a Primeira (cada atributo contém apenas valores atómicos) e Segunda (os atributos membros de uma chave são primos) formas normais e todos os atributos não primos são dependentes de uma chave de forma não transitiva. As relações encontram-se também na BCNF (o lado esquerdo de cada dependência corresponde a uma chave).

A única exceção é o conjunto de dependências correspondentes à relação Equipamento, na qual Modelo -> {Marca, NomeTipo} corresponde a uma violação da BCNF e da Terceira Forma Normal (Modelo não faz parte de uma chave).

De modo a resolver a violação das formas normais na relação Equipamento, alterou-se a classe Equipamento, passando os atributos Modelo, Marca e NomeTipo para uma classe nova Modelo, cuja chave principal é o nome do modelo (tendo em conta que este é primo). Resumidamente, o modelo relacional muda na seguinte forma:

**Equipamento**(IDEquipamento, NumeroSala -> Sala, NomeModelo -> Modelo)

Dependências funcionais:

* IDEquipamento -> {NumeroSala, NomeModelo}

**Modelo**(NomeModelo, Marca, NomeTipo -> TipoEquipamento)

Dependências funcionais:

* NomeModelo -> {Marca, NomeTipo}

Nesta forma, não existem violações da BCNF (e consequentemente das formas normais das quais é um super set [Primeira, Segunda e Terceira formas normais]).

# Implementação das Restrições

Relação **Pessoa**:

* **cartaoCidadao**: Como identifica uma e só uma pessoa, é usado como chave primária para esta relação (PRIMARY KEY);
* **nome**: Não existe uma pessoa sem nome, daí ser usada a constraint NOT NULL.
* **dataNascimento**: Tal como **nome**, não existe uma pessoa sem data de nascimento, sendo usada então a contraente NOT NULL.
* **telefone**: Por razões de organização, uma biblioteca necessita do contacto dos utilizadores e funcionários, daí a necessidade de telefone ser definido no momento do registo (NOT NULL).

Relação **Utilizador**:

* **cartaoCidadao**: Tal como na relação **Pessoa**, **cartaoCidadao** identifica uma e apenas uma pessoa, sendo assim uma chave primária (PRIMARY KEY) e uma chave estrangeira para a tabela **Pessoa** (REFERENCES Pessoa(cartaoCidadao)).

Relação **Funcionario**:

* **cartaoCidadao**: O mesmo caso que em Utilizador, cartaoCidadao serve como chave primária (PRIMARY KEY) e chave estrangeira para Pessoa (REFERENCES Pessoa(cartaoCidadao));
* **contribuinte**: O número de contribuinte de um funcionário identifica esse e apenas esse funcionário e deve ser gravado no momento de inscrição, daí ser necessário usar UNIQUE e NOT NULL, respetivamente;
* **morada**: No momento de inscrição deve ser gravado a morada atual de um funcionário, sendo necessária a restrição NOT NULL;
* **horaEntrada**: Com a inscrição de um novo funcionário, deve ser estabelecido um horário para este, sendo necessária uma restrição NOT NULL;
* **horaSaida**: O mesmo caso que **horaEntrada**, necessitando de uma restrição NOT NULL.

Relação **Reserva**:

* **idReserva**: Cada reserva deve ter um número, de modo a identificá-la devidamente, e a criação desse número deve ser feita de forma automática. Para tal, **idReserva** deve ter as restrições PRIMARY KEY e AUTOINCREMENT;
* **data**: Cada reserva deve ter uma data definida de modo a caracterizar a mesma e evitar conflitos entre reservas, daí o uso da restrição NOT NULL;
* **hora**: O mesmo caso que **data**, **hora** usa a restrição NOT NULL;
* **duracao**: Tal como **data** e **hora**, **duracao** usa a restrição NOT NULL;
* **ccUtilizador**: Cada reserva deve indicar a pessoa responsável pela mesma, para esse efeito **ccUtilizador** é uma chave estrangeira para Utilizador, sendo usada a restrição REFERENCES Utilizador(cartaoCidadao).

Relação **Sala**:

* **numero**: Cada sala, por motivos de logística, deve ter um número identificador, de modo a identificá-la melhor. Assim, **numero** é a chave principal para a relação **Sala**, tendo como restrição PRIMARY KEY;
* **capacidade**: No momento de registo de uma sala, deve-se ter em consideração o tamanho da sala, visto que não há salas “sem tamanho”, sendo necessária a restrição NOT NULL.

Relação **ReservaDeSala**:

* **idReserva**: Uma reserva de sala deve identificar a Reserva base (com informação sobre data, hora, duração e utilizador responsável), sendo **idReserva** uma chave estrangeira para **Reserva**, necessitando a restrição REFERENCES Reserva(idReserva);
* **numeroSala**: Tal como em **idReserva**, **numeroSala** precisa de identificar a sala reservada, sendo assim uma chave estrangeira, o que implica a restrição REFERENCES Sala(numero);
* Sendo a relação identificada pelos seus dois parâmetros, **idReserva** e **numeroSala** servem como chave composta principal, implicando a restrição PRIMARY KEY (idReserva, numeroSala).

Relação **Equipamento**:

* **idEquipamento**: Cada equipamento deverá ter um número identificador único. Cada número deverá fazer parte de uma sequência em conjunto com os números identificadores restantes. Para este efeito, é necessária a restrição PRIMARY KEY em conjunto com AUTOINCREMENT;
* **modelo**: Cada equipamento deverá identificar o seu modelo respetivo, tendo **modelo** como chave estrangeira para a relação **Modelo**, daí o uso da restrição Modelo(nomeModelo);
* **numeroSala**: Cada equipamento deverá também identificar a sala onde este poderá ser encontrado, logo numeroSala será uma chave estrangeira para a relação **Sala**, sendo necessária a restrição REFERENCES Sala(numero).

Relação **Modelo**:

* **nomeModelo**: Para cada modelo, o seu nome será primo e deverá identificá-lo devidamente, daí a aplicação da restrição PRIMARY KEY sobre **nomeModelo**;
* **marca**: Como não existem modelos sem marca, dever-se-á identificar a marca correspondente na inscrição na tabela **Modelo**, usando assim a restrição NOT NULL;
* **nomeTipo**: Cada modelo tem um tipo de equipamento, o qual deve ser identificado com uma chave estrangeira para **TipoDeEquipamento**, usando a restrição REFERENCES TipoEquipamento(nome).

Relação **TipoEquipamento**:

* **nome**: O nome que identifica um tipo de equipamento deve ser válido. Neste caso deverá ser um de seis: 'Leitor de Cds', 'Leitor de DVDs', 'Leitor de VHS', 'Computador', 'Projetor', 'Televisao'. Para este efeito, são necessárias as restrições PRIMARY KEY e CONSTRAINT nome\_tipo CHECK (nome = 'Leitor de Cds' OR nome = 'Leitor de DVDs' OR nome = 'Leitor de VHS' OR nome = 'Computador' OR nome = 'Projetor' OR nome = 'Televisao').

Relação **Requisicao**:

* **idRequisicao**: De modo a identificar devidamente uma requisição, é necessário um identificador primo para cada requisição, sendo aplicada a restrição PRIMARY KEY a **idRequisicao**;
* **data**: Uma requisição não pode ser feita sem ter a data da requisição, para efeitos de controlo de dias de atraso e multas, sendo necessária a restrição NOT NULL;
* **hora**: Para controlar com rigor a disponibilidade dos exemplares, dever-se-á guardar a hora na qual foi feita cada reserva, sendo necessária a restrição NOT NULL;
* **ccUtilizador**: Por cada requisição há um utilizador que a realiza, sendo necessário guardar uma referência ao mesmo, aplicando assim a restrição REFERENCES Utilizador(cartaoCidadao).

Relação **Exemplar**:

* **idExemplar**: De modo a identificar cada exemplar, é necessário um número identificador que, por coerência, deverá ser auto incrementado com cada adição à biblioteca, daí o uso das restrições PRIMARY KEY e AUTOINCREMENT;
* **possivelRequisitar**: possivelRequisitar deverá de ser um valor boolean para indicar o estado da disponibilidade de cada exemplar, sendo necessária a restrição CONSTRAINT boolean CHECK (possivelRequisitar = 0 OR possivelRequisitar = 1);
* **idSala**: Um exemplar pode ser encontrado apenas numa sala, tendo um **idSala** único que identifica a sala correspondente. **idSala** aplica a restrição REFERENCES Sala(numero);
* **idPublicacao**: A um exemplar corresponde uma publicação única, sendo que **idPublicacao** identifica uma linha na tabela **Publicacao** REFERENCES Publicacao(idPublicacao).

Relação **TipoDeManutencao**:

* **Nome**: De modo a identificar um tipo de manutenção pelo nome, o seu nome deverá ser único, daí ser necessária a restrição PRIMARY KEY.

Relação **AtoDeManutencao**:

* **ccFuncionario** Um ato de manutenção deve ser realizado por um funcionário, sendo necessário identificá-lo devidamente através do uso da restrição REFERENCES Funcionario(cartaoCidadao);
* **nomeManutencao**: Um ato de manutenção deve também identificar o tipo de manutenção a realizar, sendo que para tal é necessário aplicar a restrição REFERENCES TipoDeManutencao(nome);
* **idExemplar**: Tal como para **ccFuncionario** e **nomeManutencao**, **idExemplar** deve identificar devidamente o exemplar no qual é feito o ato, implicando o uso da restrição REFERENCES Exemplar(idExemplar);
* Como cada ato é identificado pela combinação de funcionário/tipo de manutenção/exemplar, a combinação dos atributos que os identificam deve ser única, aplicando assim a restrição PRIMARY KEY(ccFuncionario, nomeManutencao, idExemplar).

Relação **Publicacao**:

* **idPublicacao**: O número identificador de uma publicação deverá ser único de modo a identificar uma e apenas uma publicação e, por motivos de conveniência, deverá de fazer parte de uma sequência por ordem de adição à base de dados. Para tal são necessárias as restrições PRIMARY KEY e AUTOINCREMENT;
* **nome**: Como não existem publicações sem nome, **nome** deverá de ser definido no momento de adição, pelo que é necessária a restrição NOT NULL.

Relação **Livro**:

* **idPublicacao**: Como um Livro deriva de uma Publicação, o **idPublicacao** que identifica um livro, deve também identificar a publicação correspondente, assim torna-se necessário o uso das restrições PRIMARY KEY e REFERENCES Publicacao(idPublicacao);
* **editora**: Todos os livros devem ter uma editora associada, logo **editora** deve aplicar uma restrição NOT NULL;
* **edicao**: Tal como **editora**, não há livros sem um número de edição, pelo que deve ser adicionado no momento de adição à base de dados. Para além disso, um número de edição não pode ser um valor negativo (visto que o número de edição começa sempre na primeira). Assim, aplica-se as restrições NOT NULL e CHECK (edicao > 0).

Relação **Autor**:

* **idAutor**: O número identificador de um autor deve ser único, de modo a identificar apenas o autor correspondente e, por motivos de conveniência, deve seguir a ordem de adição à base de dados. Para tal, é necessário aplicar as restrições PRIMARY KEY e AUTOINCREMENT;
* **nome**: Como não existem pessoas sem nome, o nome de autor deve ser definido para um **Autor**, sendo necessária a restrição NOT NULL.

Relação **Autoria**:

* **idPublicacao**: Sendo uma relação de associação entre um livro e um autor, **Autoria** deve identificar devidamente a publicação em questão, sendo necessária a restrição REFERENCES Publicacao(idPublicacao);
* **idAutor**: Tal como para **idPublicacao**, **Autoria** deve identificar também o autor respetivo, daí o uso da restrição REFERENCES Autor(idAutor);
* Como **Autoria** corresponde a uma associação entre um livro e um autor, a combinação dos identificadores respetivos identifica uma associação **Autoria**, sendo necessária a restrição PRIMARY KEY(idPublicacao, idAutor).

Relação **Software**:

* **idPublicacao:** Sendo derivado de Publicação, Software será identificado pelo mesmo identificador que **Publicacao**, sendo assim, **idPublicacao** aplicará as restrições PRIMARY KEY e REFERENCES Publicacao(idPublicacao);
* **versao**: Semelhante à edição de um livro, a versão de software deverá de ser um valor superior a 0 e não há software sem número de versão. Para tal, são necessárias as restrições NOT NULL e CHECK(versao > 0);
* **developer**: Tal como um autor, para um determinado software, o seu developer deve ser especificado, implicando o uso da restrição NOT NULL.

Relação **Album**:

* **idPublicacao**: Sendo derivado de **Publicacao**, o identificador de **Album**  deverá de ser um identificador para **Publicacao**, implicando o uso das restrições PRIMARY KEY e REFERENCES Publicacao(idPublicacao);
* **produtor**: Para determinado álbum, o respetivo produtor deverá de ser identificado, para tal usa-se a restrição NOT NULL.

Relação **Artista**:

* **idArtista**: O identificador de um artista deverá de ser um número único que, por razões de conveniência, deverá seguir a ordem de adição na base de dados, necessitando as restrições PRIMARY KEY e AUTOINCREMENT;
* **nome**: Como não existem pessoas sem nome, um artista deverá de ter um nome associado, tendo **nome** com a restrição NOT NULL.

Relação **Interpreta**:

* **idPublicacao**: A tabela **Interpreta** indica associações entre álbuns e os respetivos artistas, logo esta tabela deve identificar devidamente o álbum em questão. Para tal, **idPublicacao** implementa a restrição REFERENCES Publicacao(idPublicacao);
* **idArtista**: Tal como para **idPublicacao**, **idArtista** deverá de identificar devidamente o artista que interpreta um determinado álbum. Assim, **idArtista** implementa a restrição REFERENCES Artista(idArtista);
* Como a relação **Interpreta** é definida pela associação entre um artista e um álbum, deve ser implementada a restrição PRIMARY KEY(idPublicacao, idArtista).

Relação **Filme**:

* **idPublicacao:** Tal como as restantes relações derivadas de **Publicacao**, o identificador para **Filme** deverá de ser um identificador para a publicação correspondente. Assim, deve-se implementar as restrições PRIMARY KEY e REFERENCES Publicacao(idPublicacao) em **idPublicacao**.
* **Realizador**: Como não existem filmes sem realizador, **Realizador** não pode ser deixado em branco, implicando a restrição NOT NULL.
* **estudio**: Tal como **Realizador**, não há um filme sem ter um estúdio associado, para tal deve-se implementar a restrição NOT NULL.

Relação **RequisicaoDeExemplar**:

* **idRequisicao**: **RequisicaoDeExemplar** corresponde a uma associação entre uma requisição e um exemplar, havendo a necessidade de identificar devidamente a requisição em questão. Para tal efeito, é necessário implementar a restrição REFERENCES Requisicao (idRequisicao);
* **idExemplar**: Tal como para **idRequisicao,** deve ser identificado devidamente o exemplar em questão, implementando em **idExemplar** a restrição REFERENCES Exemplar(idExemplar);
* Como **RequisicaoDeExemplar** é identificada pelo par requisição/exemplar, o seu identificador será a combinação dos identificadores das relações associadas, sendo necessário implementar a restrição PRIMARY KEY (idRequisicao, idExemplar).

# Interrogações (queries)

Para a terceira entrega do nosso projeto, foram selecionadas 10 interrogações, tal como pedido na descrição do mesmo. Estas interrogações tentam dar uso ao maior número possível de operadores da linguagem SQL e também têm complexidades distintas. As interrogações criadas estão descritas mais extensivamente nas próximas subsecções:

Interrogação 1

Esta interrogação pretende responder à pergunta: “Quais são os utilizadores com multas de atraso na entrega das publicações requisitadas e qual foi a multa aplicada?”. Para tal, são utilizadas diversas junções (Natural Join), de modo a conseguir juntar as informações dos utilizadores/pessoas com as requisições de exemplares. O resultado conta com os números de cartão de cidadão e o nome das pessoas, o valor da multa e o nome da publicação que a originou.

Interrogação 2

Esta interrogação responde à pergunta: “Quantas publicações existem de cada tipo?”. Na nossa base de dados, existem 4 tipos de publicações (livros, filmes, álbuns e software), divididos em classes diferentes. Esta query, utiliza o operador *Count* de modo a contar os diferentes identificadores de publicações dos vários tipos (usando *distinct*). O resultado é uma tabela com 4 colunas: a primeira contém o número de livros, a segunda contém o número de software, a terceira contém o número de álbuns e a última contém o número de filmes presentes na biblioteca.

Interrogação 3

A terceira interrogação indica a média de tempo de reservas de salas, arredondada às centésimas. São utilizados os operadores *Round* (arredondar valor) e *Avg* (média dos valores da tabela). O resultado é uma coluna com a média da duração das reservas.

Interrogação 4

A interrogação 4 lista todas as salas que nunca foram reservadas. Nesta query é utilizado o operador *In* associado a um *Not* de modo a verificar quais são as salas cujo número não aparece em nenhuma reserva. O resultado são as salas, ou seja, uma tabela com o número, tipo e capacidade, que nunca foram reservadas.

Interrogação 5

Na quinta interrogação responde-se à pergunta: “Quais são os funcionários que praticaram um ato de manutenção, qual foi esse ato e quantas vezes o exerceram?”. Para isto é feita uma junção (*Natural Join*) de todos os funcionários participantes num ato de manutenção e o numero de vezes que cada um foi realizado. O resultado é uma tabela como cartão de cidadão do funcionário, o nome da manutenção e o número de vezes que foi realizada.

Interrogação 6

A sexta interrogação pretende mostrar o top 10 de publicações mais requisitadas. Surge novamente o operador de junção natural, juntamente com um *Group By* que agrupa requisições com o mesmo identificador e um *Order By* com *Limit* de 10 tuplos de modo a listar somente10 publicações. O resultado é uma tabela com o nome e número de requisições de cada publicação. Caso não haja 10 publicações requisitadas, apenas são listadas aquelas que foram requisitadas.

Interrogação 7

Esta interrogação pretende responder à pergunta: “Para cada tipo de publicação, qual é a publicação mais requisitada e o respetivo criador?”. Na resolução desta query é utilizada uma vista intitulada publicacoesRequisitadas que seleciona o identificador de publicação, o nome e o número de requisições. A partir desta vista são realizadas junções com as tabelas dos tipos de publicações, de modo a obter a publicação mais requisitada. Após obter estes dados, estes são juntados através do operador *Union* resultando uma tabela com o nome do criador e o número de requisições dessa publicação.

Interrogação 8

A interrogação 8 responde à pergunta: “Quais são as salas reservadas, os responsáveis pela reserva e a sua duração?”. Para além de uma *Natural Join* é utilizado outro tipo de junção do sqlite (*Join…On*). O resultado é uma tabela com o número da sala, o cartão de cidadão e o nome do responsável pela reserva e a sua duração.

Interrogação 9

Esta interrogação lista todos os utilizadores e o seu número de requisições. Para tal, é utilizada uma junção entre a tabela Pessoa e Requisição. O resultado é uma tabela com os nomes e o número de requisições realizadas.

Interrogação 10

Com esta interrogação pretende-se listar todos os jovens utilizadores, com idade compreendida entre os 18 e 25 anos. Nesta interrogação são utilizadas as funções do sqlite strftime(), que converte uma data num certo tipo dado, e cast(), que no nosso caso transforma a string resultante num inteiro. É utilizado também o operador *Between* que é utilizado para encontrar as idades no intervalo desejado. O resultado da query é uma tabela com o nome do utilizador e a sua idade.

# Gatilhos (*triggers*)

Para além das 10 interrogações, foram selecionados 3 gatilhos que permitem a melhor manutenção da base de dados. Cada gatilho tem uma função diferente, sendo que um implementa uma restrição (*Not Null*). Todos os gatilhos estão descritos nas próximas subsecções:

Gatilho 1

O primeiro gatilho tem como função o preenchimento automático de valores, ou seja, implementa uma restrição do tipo *Not Null*. Este *trigger* faz com que cada tuplo introduzido na tabela AtoDeManutencao no qual o funcionário não é especificado ou é introduzido um valor *NULL*, obtenha o valor do cartão de cidadão do funcionário (identificador do funcionário) que realizou menos atos de manutenção até à data onde é introduzido o valor.

Gatilho 2

O segundo gatilho tem como função a verificação de valores na base de dados, mais especificamente na tabela Exemplar. O segundo *trigger* impede a requisição de um exemplar quando este não se encontra disponível para tal.

Gatilho 3

O terceiro gatilho tem como objetivo o cálculo de um valor. Foi adicionado um valor à relação Publicacao, que indica a quantidade de exemplares existentes de cada publicação. Este valor é incrementado sempre que é adicionado um exemplar desta publicação.

Nota: este *trigger* deve ser criado antes da base de dados ser povoada, de modo a ter todos os valores corretos quando o *script gatilho3\_verifica.sql* é corrido.