2018/19



BRUNO FILIPE CARDOSO MICAELO – UP201706044

MIGUEL DELGADO PINTO – UP201706156

PEDRO MIGUEL RODRIGUES FERRAZ ESTEVES – UP201705160

Gestão de Recursos de Biblioteca

Base de Dados

# 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.

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.

Caso um utilizador esteja atrasado na devolução de um exemplar, a classe de associação **Atraso** guarda informação sobre os dias de atraso e a multa.

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)

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)

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)

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)

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)

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)

Nome -> {Proposito}

A chave principal para TipoEquipamento é Nome.

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

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)

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)

IDPublicacao -> {Nome, Genero, IdadeMinima}

A chave principal da relação Publicacao é IDPublicacao.

Livro(IDPublicacao -> Publicacao, Editora, Edicao)

IDPublicacao -> {Editora, Edicao}

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

Autor(IDAutor, Nome)

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)

IDPublicacao -> {Versao, Developer}

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

Album(IDPublicacao -> Publicacao, Produtor)

IDPublicacao -> {Produtor}

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

Artista(IDArtista, Nome)

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)

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}

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 (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) e 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 (Modelo não faz parte de uma chave).

De modo a resolver a violação da BCNF 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)

IDEquipamento -> {NumeroSala, NomeModelo}

Modelo(NomeModelo, Marca, NomeTipo -> TipoEquipamento)

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** INTEGER REFERENCES Sala(numero) ON DELETE CASCADE ON UPDATE CASCADE,
* **idPublicacao** INTEGER REFERENCES Publicacao(idPublicacao) ON DELETE CASCADE ON UPDATE CASCADE,

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** INTEGER NOT NULL REFERENCES Funcionario(cartaoCidadao) ON DELETE CASCADE ON UPDATE CASCADE,
* **nomeManutencao** TEXT NOT NULL REFERENCES ipoDeManutencao(nome) ON DELETE CASCADE ON UPDATE CASCADE,
* **idExemplar** INTEGER NOT NULL REFERENCES Exemplar(idExemplar) ON DELETE CASCADE ON UPDATE CASCADE,
* PRIMARY KEY(ccFuncionario, nomeManutencao, idExemplar)

Relação **Publicacao**:

* **idPublicacao** INTEGER PRIMARY KEY AUTOINCREMENT,
* **nome** TEXT NOT NULL

Relação **Livro**:

* **idPublicacao**: Como um Livro entra na categoria de PublicaçãoREFERENCES Publicacao(idPublicacao) ON DELETE CASCADE ON UPDATE CASCADE,
* **editora** TEXT NOT NULL,
* **edicao** INTEGER CHECK (edicao > 0) NOT NULL

Relação **Autor**:

* **idAutor** INTEGER PRIMARY KEY AUTOINCREMENT,
* **nome** TEXT NOT NULL

Relação **Autoria**:

* **idPublicacao** INTEGER NOT NULL REFERENCES Publicacao(idPublicacao) ON DELETE CASCADE ON UPDATE CASCADE,
* **idAutor** INTEGER NOT NULL REFERENCES Autor(idAutor) ON DELETE CASCADE ON UPDATE CASCADE,
* PRIMARY KEY(idPublicacao, idAutor)

Relação **Software**:

* **idPublicacao** INTEGER PRIMARY KEY NOT NULL REFERENCES Publicacao(idPublicacao) ON DELETE CASCADE ON UPDATE CASCADE,
* **versao** FLOAT NOT NULL CHECK(versao > 0),
* **developer** TEXT NOT NULL

Relação **Album**:

* **idPublicacao** INTEGER PRIMARY KEY NOT NULL REFERENCES Publicacao(idPublicacao) ON DELETE CASCADE ON UPDATE CASCADE,
* **produtor** TEXT NOT NULL

Relação **Artista**:

* **idArtista** INTEGER PRIMARY KEY AUTOINCREMENT,
* **nome** TEXT NOT NULL

Relação **Interpreta**:

* **idPublicacao** INTEGER NOT NULL REFERENCES Publicacao(idPublicacao) ON DELETE CASCADE ON UPDATE CASCADE,
* **idArtista** INTEGER NOT NULL REFERENCES Artista(idArtista) ON DELETE CASCADE ON UPDATE CASCADE,
* PRIMARY KEY(idPublicacao, idArtista)

Relação **Filme**:

* **idPublicacao** INTEGER PRIMARY KEY NOT NULL REFERENCES Publicacao(idPublicacao) ON DELETE CASCADE ON UPDATE CASCADE,
* **realizador** TEXT NOT NULL,
* **estudio** TEXT NOT NULL

Relação **RequisicaoDeExemplar**:

* **idRequisicao** INTEGER NOT NULL REFERENCES Requisicao(idRequisicao) ON DELETE CASCADE ON UPDATE CASCADE,
* **idExemplar** INTEGER NOT NULL REFERENCES Exemplar(idExemplar) ON DELETE CASCADE ON UPDATE CASCADE,
* PRIMARY KEY (idRequisicao, idExemplar)