Book Sharing

Relatório de Projeto no âmbito da Unidade Curricular de **Bases de Dados**

Mestrado Integrado em Engenharia Informática e Computação

Ano letivo 2017/2018

| Grupo 4 da Turma 3 |

César Medeiros | 201605344

Margarida Silva | 201606214

Pedro Costa | 201605125

Índice

Descrição do contexto

A nossa base de dados tem como fim o suporte de uma aplicação de **Book Sharing**.

Na aplicação os utilizadores podem partilhar e requisitar livros, trocando mensagens entre si para facilitar o processo de troca.

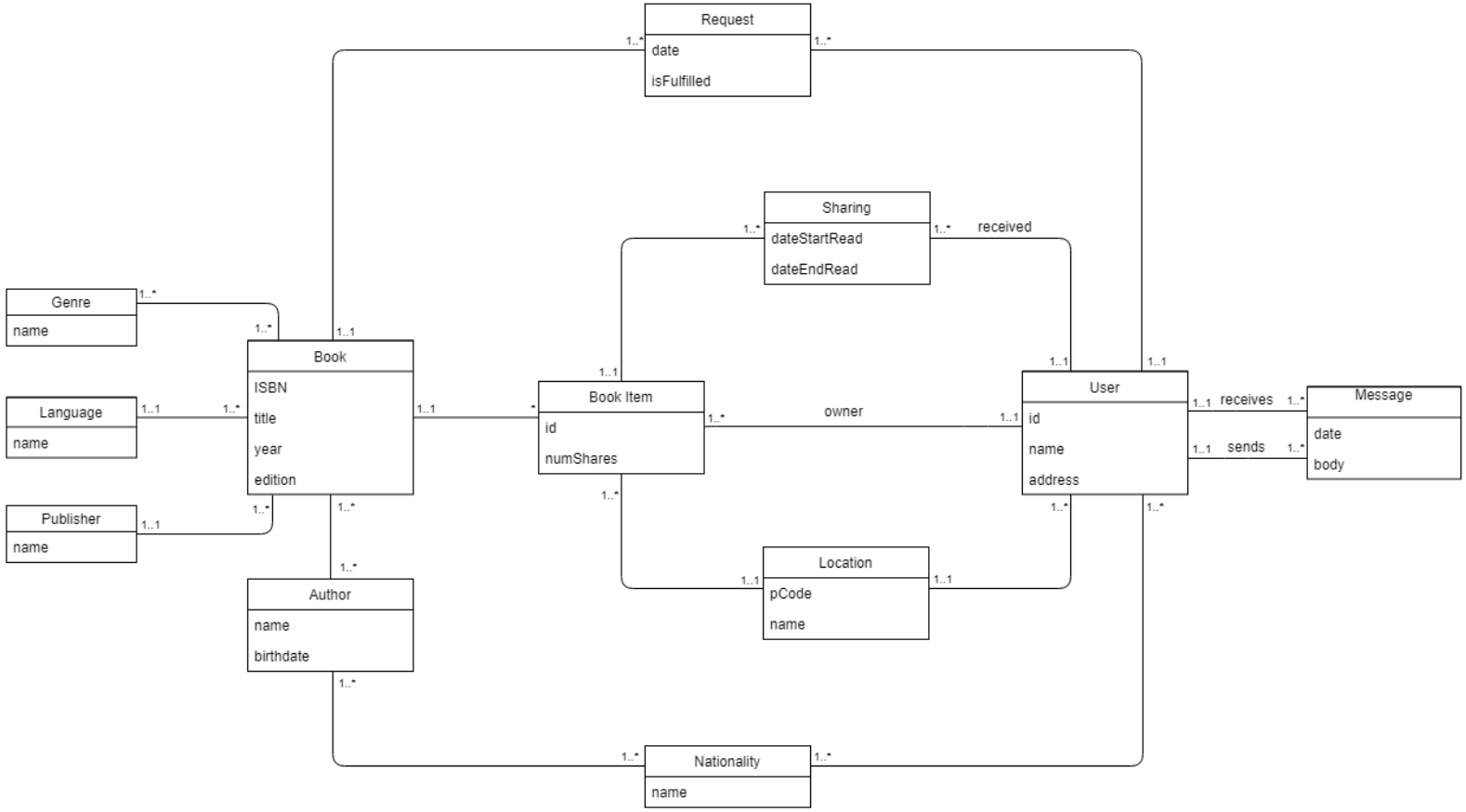
Um utilizador acede à aplicação e acede a um conjunto de Book Items que estão disponíveis para lhe serem emprestados. Cada Book Item está associado a um Book, que por sua vez está associado ao autor, linguagem, género e editora.

Por outro lado, é oferecida a possibilidade do utilizador inserir Requests de livros. Estes servem apenas para dar a indicação de que um determinado utilizador anda à procura de um determinado livro e ainda não conseguiu agendar uma troca (seja por falta de stock na base de dados (ninguém tem o livro), incompatibilidades de tempo/localização com os users, etc). Quando vê a sua necessidade do livro chegar ao fim, preenche o campo isFulfilled – seja porque já trocou o livro, ou simplesmente já não tem mais a necessidade de obter aquele livro em específico.

No processo de Sharing, é associado um Book Item e um User a quem vai ser emprestado o livro, sendo guardada a data de troca, e a data do fim da leitura. Este último atributo está nulo enquanto o empréstimo decorre, sendo apenas preenchido quando o user dá a leitura por terminada. Nessa altura, o Book Item volta a estar disponível para que outros utilizadores o possam pedir emprestado também. O processo de partilha é acordado via mensagens entre utilizadores. Como cada user tem uma localização associada, é possível que os utilizadores vão procurando uns aos outros em localizações próximas, estabelecendo contactos.

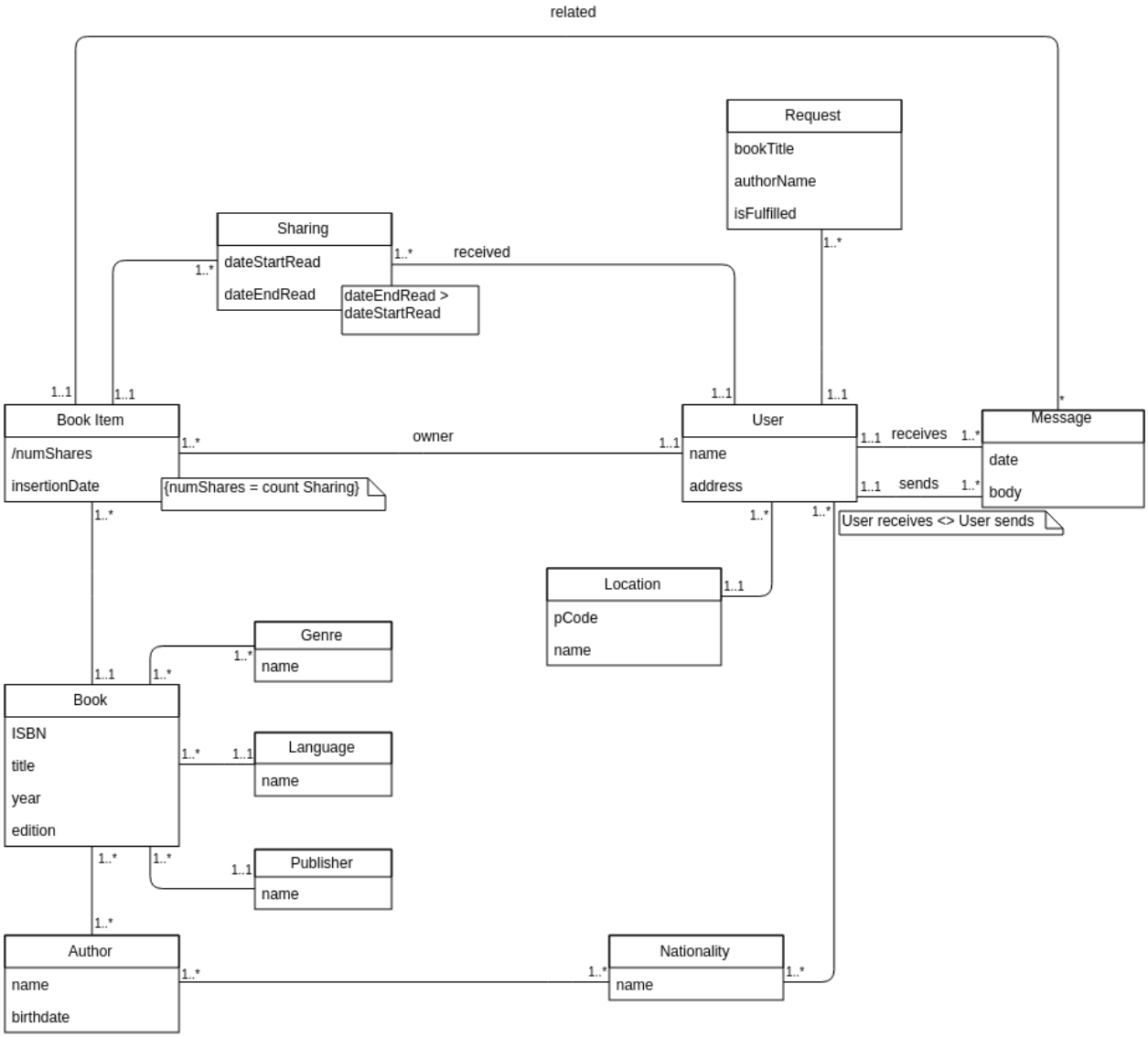
Diagrama de Classes UML

Primeira Abordagem



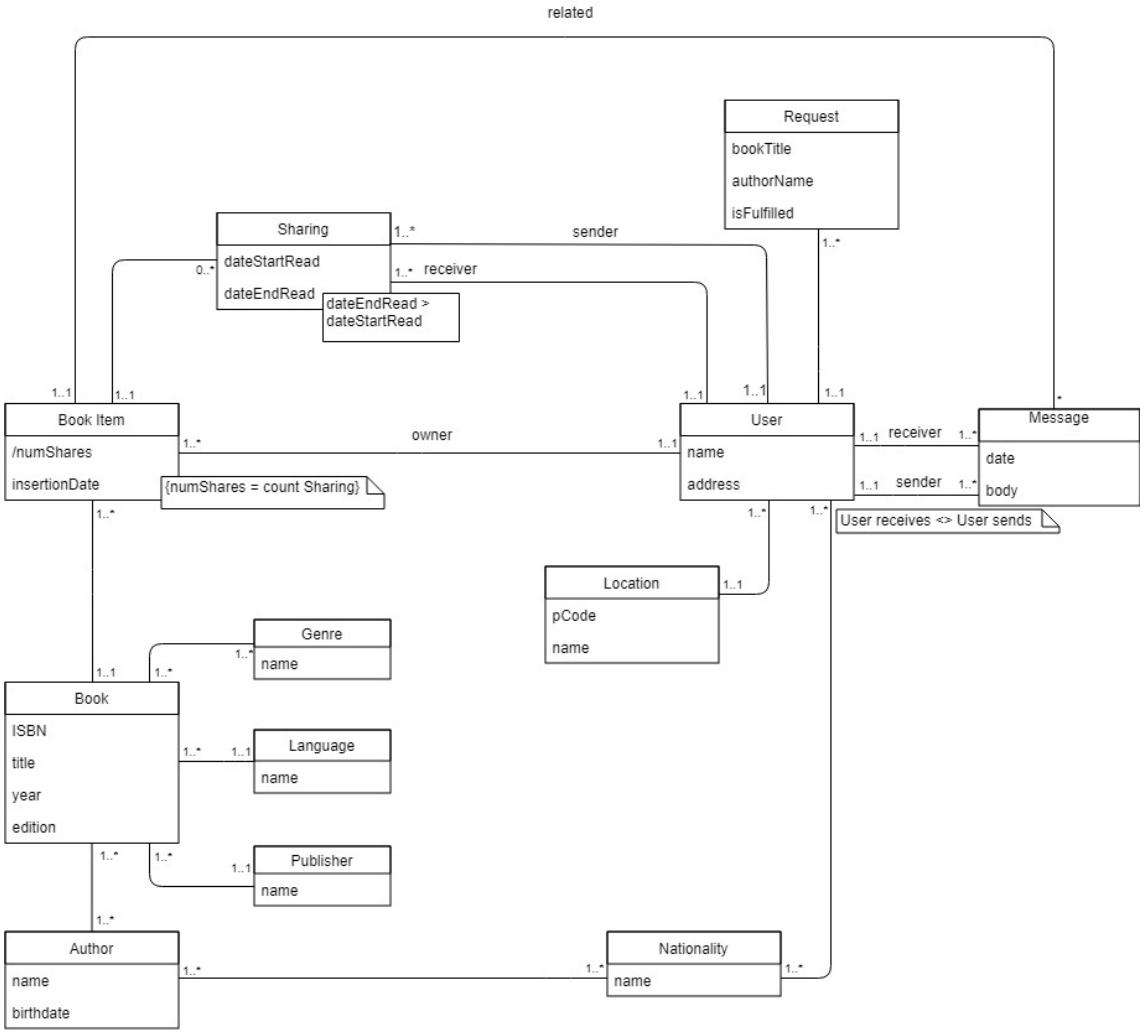
Segunda Abordagem

Após a primeira entrega, discutimos uma nova solução com a professora, e concluímos a adição de uma relação entre *BookItem* e *Message*, de forma a que as mensagens sejam relativas a um determinado exemplar.



Terceira Abordagem

Aquando desta última entrega, achámos por bem adicionar uma relação entre *Sharing* e *User*, *receiver*, por forma a conseguir ter um registo dos *owners* antigos de um livro (após as sucessivas *Sharings*).



Modelo relacional e Dependências Funcionais não Triviais

> ***Genre*** (id, name)

i. {id}→{name};

ii. {name}→{id};

> ***Language*** (id, name)

i. {id}→{name};

ii. {name}→{id};

> ***Publisher*** (id, name)

i. {id}→{name};

ii. {name}→{id};

> ***Nationality*** (id, name)

i. {id}→{name};

ii. {name}→{id};

> ***Book*** (id, ISBN, title, year, edition, language → Language, publisher → Publisher)

i. {id}→{ISBN};

ii. {ISBN}→{title, year, edition, language, publisher};

iii. {ISBN} → {id};

> ***BookGenre*** (idB → Book, idG → Genre)

> ***BookAuthor*** (idA → Author, idB → Book)

> ***Author*** (id, name, birthdate)

i. {id} → {name, birthdate};

> ***AuthorNationality*** (author→ Author, nationality→ Nationality)

> ***Location*** (id, pCode, name)

i. {id} → {pCode};

ii. {pCode} → {name};

iii. {pCode}→{id};

> ***User*** (id, name, address, location→ Location)

i. {id} → {name, address, location};

> ***UserNationality*** (user→ Nationality, nationality→ Nationality)

> ***BookItem*** (id, numShares, insertionDate, book→Book, owner→User)

i. {id} → {numShares, insertionDate, book, owner};

> ***Sharing*** (book→ BookItem, receiver → User, startDate, endDate)

i. {startDate, book, receiver}→{endDate};

ii. {endDate, book}→{startDate, receiver};

> ***Request*** (id, bookTitle, authorName, isFulfilled, request→ User)

i. {id} → {bookTitle, authorName, isFulfilled, request};

> ***Message*** (id, date, body, receiver → User, sender→ User,context→BookItem)

i. {id} → {date, body, receiver, sender, context};

ii. {date, sender} → {id, body, receiver, context];

Análise de Formas Normais

As violações à terceira forma normal e à forma normal de Boyce-Codd estão presentes nas relações:

Book

Nas dependência ii e iii, o lado esquerdo, {ISBN}, não é uma super-chave[1], logo a relação não se encontra na forma normal Boyce-Codd.

Ainda na dependência ii, o lado direito, {title, year, edition, language, publisher}, contém atributos não primos (a chave candidata é apenas {id}).

Pode-se assim concluir, tendo em conta os dois pontos anteriores que a relação não se encontra na terceira forma normal.

> *Location*

Motivo análogo ao enunciado em Book.

Todas as outras relações cumprem todas as condições das formas normais.

A relação Sharing está na terceira forma normal e Boyce-Codd pois o lado esquedo das suas dependências funcionais {startDate, book, receiver}[1] {endDate, book}[2] são super-chaves da relação. O mesmo motivo se aplica à relação Message, com {id} e {date, sender}.

Nas relações Genre, Language, Publisher e Nationality, como tanto name como id são super-chaves, também estão na BCNF, e, assim, na 3NF.

As restantes como só possuem uma dependência funcional, o lado esquerdo será obrigatoriamente uma chave, neste caso, chave candidata.

[1] $\{startDate, book, receiver\}^+ = \{startDate, book, receiver, endDate\}$

[2] $\{endDate, book\}^+ = \{startDate, book, receiver, endDate\}$

Restrições

Para assegurar consistência e correção na inserção de dados foram implementadas restrições em todas as tabelas.

Lista

Para evitar a repetição da mesma informação em vários pontos desta seção, assume-se que:

⇒ Em cada relação que possui o atributo id, como não podem existir duas instâncias com o mesmo id (p.ex, dois utilizadores com o mesmo id), este mesmo foi definido como *primary key*. Tais relações são: *Genre, Language, Publisher, Nationality, Book, Author, Location, User, BookItem, Request, Message.*

⇒ Sempre que é dito que um atributo tem de estar definido (p.ex, o *name* de um utilizador), foi utilizada a restrição not null.

Nas relações Genre, Language, Publisher e Nationality o atributo *name* tem de se encontrar definido e é único, restrição unique; não faria sentido existir uma linguagem não definida.

Em Book o ISBN é único e tem de estar definido. O *title*, *year*, e *edition* também têm de estar definidos, para assim não ocorrerem problemas na pesquisa de livros. A *language* e *publisher* de um *book* são chaves estrangeiras que apontam para as relações previamente enunciadas. Proíbe-se a eliminação de uma *publisher* ou *language* se existir pelo menos um livro associado. Nas chaves estrangeiras foram aplicadas as restrições de *on delete restrict.*

Em relação a *BookGenre*, *BookAuthor*, *AuthorNationality* e *UserNationality* foram aplicadas as mesmas restrições. São relações constituídas apenas de duas chaves estrangeiras, por exemplo, em *BookAuthor* para *Book* e para *Author*. O par constituído pelas chaves tem de ser único, se tal não acontecesse seria apenas repetição de informação. Como em *Book*, é impossível eliminar uma relação, se esta estiver referenciada por uma qualquer das referidas em cima. Assim, em cada relação, a *primary key* foi definida como o par das duas chaves estrangeiras e em cada chave estrangeira foi adicionada a restrição *on delete restrict*.

O nome de cada *Author* e *User* tem de estar definido, tal como o *address* em *User.* Cada utilizador possui uma *location* definida que é uma chave estrangeira para *Location*. Não é possível eliminar uma *Location* enquanto existir pelo menos um *User* a ela associado. Para tal, foi usada a restrição *on delete restrict.*

Em *BookItem,* a sua *insertionDate* é definida no momento em que entra para a base de dados. A consistência do atributo derivado *numShares* será garantida ao ser incrementada a cada nova inserção de *Sharing* com esse mesmo *BookItem.* Como BookItem é um exemplar de um *Book* pertencente a um *User,* contém uma chave estrangeira para *Book* e outra para *User.* Não é possível eliminar um *Book* se existe pelo menos um exemplar, garantido pela restrição *on delete restrict*. No entanto, quando um *User* sai da base de dados, toda a sua informação é apagada e assim os seus exemplares também terão de ser eliminados, com *on delete cascade*.

Em *Sharing*, ao momento da partilha em si, a *startDate* será definida, ou seja, estará sempre definida. No entanto, enquanto o *User* que recebe o livro não acabar a leitura, a *endDate* estará a *null.* Assim é possível verificar quais os livros que estão a ser lidos, a qualquer momento.

Este atributo será depois definido pelo mesmo *User* quando acabar de ler. Como é óbvio, a *endDate* será sempre superior ou igual (caso ele acabe de ler no mesmo dia que requisitou). Isto é imposto pela restrição de *check*. A uma *Sharing* está também associado um *Book*, livro a ser partilhado, e um *User* que recebe o *Book,* denotado por *receiver.* Ambos são uma chave estrangeira para *bookItem e User*, respetivamente. Quando um *bookItem* é eliminado, todas as suas partilhas são eliminadas. Quando um *User* for removido as partilhas mantêm-se pois envolve 2 users e o que não foi eliminado poderá querer ver todas as partilhas já feitas. O *user* eliminado é colocado a null na partilha. Este comportamento é obtido com a restrição de on delete cascade em book e on delete set null em receiver.

Não podem existir duas partilhas com a mesma *startDate*, o mesmo *book* e o mesmo *receiver.* A *primary key* da relação é então o conjunto {*startDate, book, receiver}.* Também não poderão existir duas partilhas onde o par *endDate* e *book* são iguais, garantido tal com a restrição unique do par.

Nesta relação não foi inserido como chave primária um atributo id pois como a relação *Sharing* não é refênciada através de uma chave estrageira por nenhuma outra relação não existe a necessidade de um atributo adicional.

Nos *Requests*, os *users* podem escrever ou o título ou o autor do livro que pretendem ler, mas que ainda não existe na base de dados. São apenas obrigados a especificar pelo menos um deles, isto pois o *user* pode não estar certo do título do livro mas sabe quem é o autor. Tal pode ser verificado com uma restrição check onde é verificado se um ou outro não são null. Tanto o título como o nome do autor não podem ter mais do que 40 caracteres; caso fosse, o mais provável seria que o utilizador estava apenas a introduzir "lixo" que não deve ser guardado. É usada então a restrição de check na length dos atributos. O *request* é também feito por um *requester*, que é uma chave estrangeira para um *user*. Quando a informação do *user* é atualizada a informação no *request* também deve ser e quando um *user* é eliminado, todos os seus *request* também devem ser eliminados. Isto é obtido com a restrição na chave estrangeira de *on delete cascade.* Um *request* tem também outro campo, isFulfilled, que indica se o *request* foi ou não respondido. Este campo estará por predefinição a 0 até que o seja respondido, onde o utilizador o atualizará para 1.

Por fim, a relação *Message*. Cada *message* terá uma data associada e um *body* cujo conteúdo não pode ser nulo e não pode exceder os 500 caracteres, verificável com a restrição *not null e check* para a length. Terá associado um remetente, *send*er, e um recetor, *receiver,* que são chaves estrangeiras para *User.* Quando um dos users envolvidos é eliminado, todas as mensagens referentes a ele serão apagadas, dado que já não existe nenhum interesse em guarda tal informação; obtido com a restrição de *on delete cascade.* Uma mensagem estará sempre no contexto de uma troca de um livro que o *user receiver* tem. Assim, existe também uma chave estrangeira para esse mesmo livro, *bookItem.* As restrições aplicadas aqui são iguais às dos *users.*

Interrogações

Segue-se a lista de interrogações que elaboramos:

1. Tempo Médio de Empréstimo

Devolve o tempo médio de um empréstimo. Por tempo entenda-se o intervalo entre a troca física do livro e a nova colocação do livro na base de dados após a leitura de quem o recebeu.

2. Ranking de *Users* mais ativos

Lista os utilizadores mais ativos da aplicação. Um utilizador considera-se ativo quando partilhou vários livros na rede.

3. Pessoas com gostos em comum (pelo menos já leram dois livros do mesmo género)

Relaciona pessoas com outras pessoas com gostos literários similares. No nosso caso, consideramos que duas pessoas têm gostos similares se tiverem lido pelo menos 2 livros de género literários iguais.

4. Livros recomendados

Recomenda alguns livros a um *User* baseado nos autores que este já leu previamente. Notar que não são contabilizados os autores livros dos quais este é *owner*, pois consideramos que nada garante que os livros que são inseridos por um User são (ou não!) da sua preferência.

5. Autores favoritos de um *User*

Mostra os autores favoritos de um utilizador, baseado em quantos livros já leu dele/a.

6. Número de *Users* por *Location*

Número de utilizadores que se encontram numa determinada localização.

7. Livros Disponíveis

Mostra todos os livros disponíveis para partilha na aplicação. Por livro disponível entende-se todo aquele que não está numa *Sharing* ativa (data de fim ainda por definir).

8. Users Responsivos a Mensagens

Lista os Users por ordem crescente de tempo médio que demoram a responder a uma mensagem, mostrando-o.

9. Autores favoritos

Lista os 3 autores favoritos de todos os utilizadores da aplicação.

10. Requests que foram cumpridos e quem partilhou o livro que possibilitou.

Lista os request que já foram cumpridos e quem partilhou o livro pretendido ao User que fez o request (no caso de ter ocorrido).

Gatilhos

Segue-se a lista de gatilhos que elaboramos:

1. validateSharing

A cada inserção de Sharing, colocar em *sender* o dono atual do BookItem em questão, atualizar o atributo *owner* deste, bem como o seu *numShares*.

2. verifyMessage

A cada inserção de Message, verificar se o dono do *BookItem* desta coincide com o *receiver* da mensagem. Caso contrário, lançar uma mensagem de erro. De notar que não é necessário o teste de o receiver ser diferente do sender pois essa verificação já é feita na criação da base de dados (create.sql), com "check(receiver <> sender)".

3. insertSharing

A cada inserção de *Message* em que o corpo do texto é do tipo "[Share][Complete]", é verificado se o *sender* desta é o *owner* do *BookItem* em questão. Se não for, é lançada uma mensagem de erro. Se for, é criada uma nova Sharing com sender, receiver e startDate adequados, e endDate a NULL. Se, por ventura, a endDate da Sharing anterior do livro não tiver sido colocada pelo (antigo) owner, esta é atualizada para a data da mensagem, também.