Capítulo 4

Aplicação

4.1 - Base de Dados ------------------------------------------------- 2/3

4.1.1 - Retirar da web -------------------------------------------- 4

4.1.2 - Criar a base de dados --------------------------------- 6

4.1.3 - Inicializar a base de dados ------------------------- 0

4.2 - Estrutura da aplicação ------------------------------------ 2/3

4.2.1 - Book.java --------------------------------------------------- 4

4.2.2 - Library.java ------------------------------------------------ 6

4.2.2.1 - Testar a Library ----------------------------------- 8

4.2.3 - Category.java --------------------------------------------- 6

4.2.4 - Carregar conteúdo da base de dados ---------- 0

4.2.4.1 - Testar o carregamento do conteúdo -------- 8

4.3 - Bloom Filter ---------------------------------------------------- 2/3

4.3.1 - Testar o Bloom Filter --------------------------------- 4

4.4 - MinHash ---------------------------------------------------------- 2/3

4.4.1 - Hash.java ---------------------------------------------------- 4

4.4.2 - MathWorksFunctions.java ------------------------- 6

4.4.3 - Similaridade e Distância de Jaccard ---------- 0

4.4.3.1 - Testar a similaridade -------------------------- 8

4.4.4 - Testar o Min Hash ------------------------------------- 0

4.5 - Menu ---------------------------------------------------------------- 2/3

4.5.1 - Listar Livros ----------------------------------------------- 4

4.5.1.1 - Listar todos os livros ---------------------------- 4

4.5.1.2 - Listar livros por categoria ---------------------- 5

4.5.2 - Pesquisar Livros ----------------------------------------- 6

4.5.2.1 - Verificar existência de livro ------------------- 6

4.5.2.2 - Pesquisar livros com títulos parecidos ----- 7

4.5.3 - Pesquisar Autores -------------------------------------- 0

4.5.3.1 - Verificar se autor existe ------------------------ 0

4.5.3.2 - Pesquisar livros de um autor ----------------- 0

4.5.4 - Administração ------------------------------------------- 7

4.5.4.1 - Adicionar Livro ----------------------------------- 8

4.5.4.2 - Remover Livro ------------------------------------ 8

4.5.4.3 - Requesitar Livro ---------------------------------- 9

4.5.4.4 - Devolver Livro ----------------------------------- 10

4.5.5 - Abandono da ação ----------------------------------- 11

**4.5 Menu**

A interface do menu da aplicação é feita através do *Terminal*. O utilizador corre o ficheiro **RunLibrary.java** para iniciar o programa, através dos seguintes comandos:

**1.** Modificar o *$CLASSPATH* do *Terminal* para poder aceder a uma biblioteca externa do java:

$ export CLASSPATH=.:/path/to/this/folder/library-management/lib/json- simple-1.1.jar

**2.** Compilar todos os ficheiros do java para “construir” o programa:

$ javac \*.java

ou, caso não seja feito o passo **1.**:

$ javac -cp /path/to/this/folder/library-management/lib/json-simple-1.1.jar \*.java

**3.** Correr o ficheiro **RunLibrary.java**:

$ java RunLibrary.java

**RunLibrary.java**

**|** **static String** libName

**|** **static BloomFilter** bm

**|** **static Library** lib

**|** **static Hash[]** listHash

**|** **static MinHash** minHash

**|** **static List<Book>** listOfBooks

**|** **public static void** displayMenu()

**|** **public static void** main()

**|** **public static double** similarityValue()

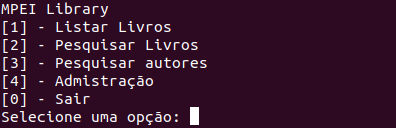
**|** **public static int** getIntersections()

**|** **public static int[]** getMinHashes()

**|** **public static void** fillHashList()

**|** **public static void** filterBooksByCategory()

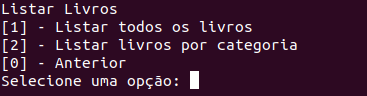
Esta *Class* possui uma main, onde vai mostrar um menu inicial, criado pela função **RunLibrary.displayMenu()**. Neste menu o utilizador pode escolher uma de várias opções, criando um input através do teclado do número da opção, que vai ser lido através de um **Scanner**. Através de um *switch-case* do input do utilizador*,* é escolhido o que fazer.



1. Screenshot do menu inicial

**4.5.1 Listar Livros**

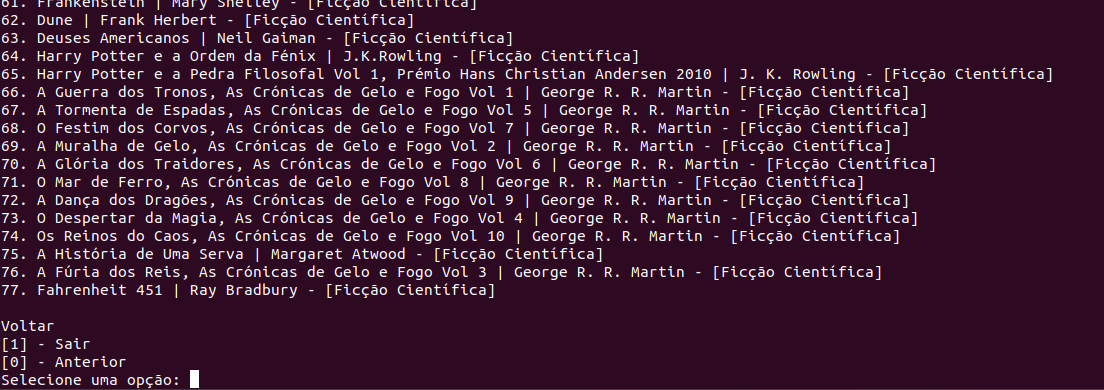
Neste menu, o utilizador o utilizador pode escolher entre duas opções: Listar todos os livros existentes no acervo, no ecrã, ou expecificar uma categoria de livros, e depois listar todos os livros dessa categoria essa categoria.



2. Screenshot do menu Listar Livros

4.5.1.1 Listar todos os livros

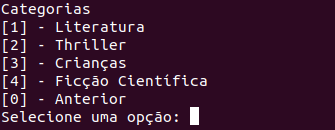
Para a listagem de todos os livros, é feita apenas uma iteração pela **RunLibrary.listOfBooks**, e é feito um print de cada elemento do seu conteúdo, que é do tipo **Book**.



3. Screenshot da listagem de todos os livros

4.5.1.2 Listar livros por categoria

Para a listagem dos livros por categoria, é dado ao utilizador um novo menu, onde este pode escolher qual categoria quer ver e depois é feito o display. Mediante a escolha do utilizador, são filtrados todos os livros que apresentam a categoria indicada e é feito um print dos mesmos. A filtragem usa a função **RunLibrary.filterBooksByCategory()**, que usa a função **MinHash.getMinHashes()** e **MinHash.similarityValue()** para perceber, das categorias existentes, qual a escolhida pelo utilizador~~.~~



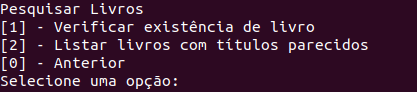
5. Screenshot da listagem dos livros de Literatura

4. Screenshot da escolha das categorias



**4.5.2 Pesquisar Livros**

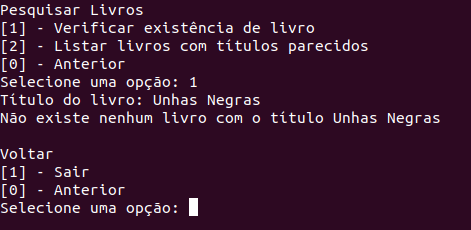
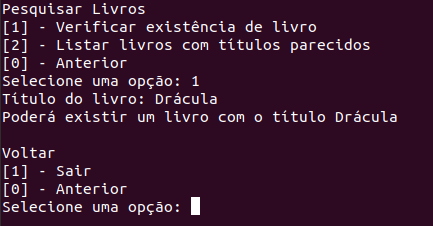
Neste menu, o utilizador pode escolher entre duas opções: Verificar a existência de um livro no acervo, que é feita recorrendo à *Class* **BloomFilter**, ou então pesquisar por todos os livros que tenham um nome similar ao nome que o utilizador passe na consola, sendo que a verificação de similaridade entre a *String* do utilizador, e a *String* referente ao nome de cada livro do acervo, é feita através do cálculo da distância de jaccard entre os arrays de hashes mínimas de cada *String*, que são obtidas através da *Class* **MinHash**.



6. Screenshot do menu Pesquisar Livros

4.5.2.1 Verificar a existência de livro

Para verificar a existência de um livro, o utilizador é, primeiramente, convidado a escrever o título de um livro. Dentro da *Class* **RunLibrary** são inseridos os títulos de todos os livros do acervo no **BloomFilter**, através da função **BloomFilter.insert()** e, por fim, é verificado se a *String* passada pelo utilizador pertence ao **BloomFilter**, através da função **BloomFilter.isMember()**. Se esta função retornar *true*, então existe a possibilidade de o título dado pelo utilizador ser o título de algum livro do acervo. Se retornar *false*, então é certo que não existe nenhum livro com esse título, no acervo.

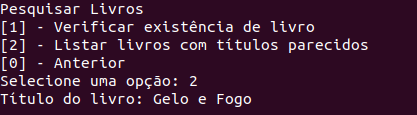


8. Screenshot da possivel existência do livro

7. Screenshot da não existência do livro

4.5.2.2 Listar livros com título parecidos

A listagem de livros com títulos parecidos é feita através de duas funções presentes na *Class* **RunLibrary**, que componhem todo o processo de obter as hashes mínimas e verificar a similaridade através da distância de jaccard. A *String* que o utilizador vai passar na consola vai passar como agumento na função **RunLibrary.getMinHashes()**. Esta função vai pegar nessa *String*, vai criar vários *shingles* através da função **MinHash.makeShingles()**, vai transformar os shingles em hashes, através da função **MathWorksFunctions.string2hash()**, vai criar centenas de hashes para cada shingle através da função **MinHash.getHashesForSingle()** e por fim vai retornar um array de hashes mínimas, de todas as hashes criadas anteriormente, através da função **MinHash.minHashes()**. Depois de este processo ser repetido para todos os livros do acervo, é calculada a distância de jaccard entre as hashes mínimas da *String* do utilizador e dos títulos dos livros. A função usada para o cálculo é a **RunLibrary.similarityValue()**, que vai calcular as interseções de ambos os arrays de hashes e vai dividir pela união dos mesmos. Por vim, se o resultado desta similaridade for favorável, então significa que em princípio os títulos são parecidos e é feito um print do livro.



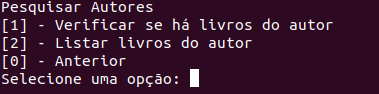
9. Screenshot da escolha da *String* pelo utilizador



10. Screenshot do resultado da pesquisa com a string Gelo e Fogo

**4.5.3 Pesquisar Autores**

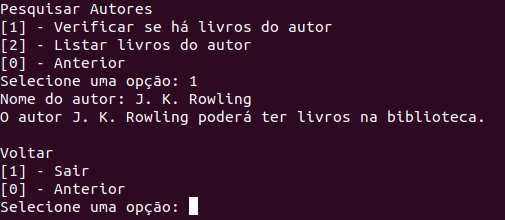
Na pesquisa de autores, é da a possibilidade de o utilizador escolher entre verificar se existem livros de um autor em específco no acervo, e depois, caso exista, o utilizador tem a possibilidade de ver todos os livros desse autor.



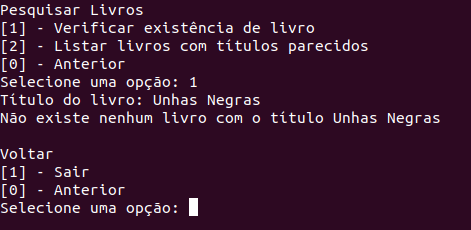
11. Screenshot do menu da pesquisa de autores

4.5.3.1 Verificar se há livros do autor

Nesta opção, é pedido ao utilizador para escrever o nome de um autor e, à semelhança do tópico **4.5.2.1 Verifica a existência de livro**, é usado o **BloomFilter** para verificar se um autor poderá ter livros seus na biblioteca. Caso isto seja verdade, o nome do autor é guardado como objeto da *Class* **RunLibrary**, para ser usado na opção referente ao tópico seguinte **4.5.3.2 Listar livros do autor**.



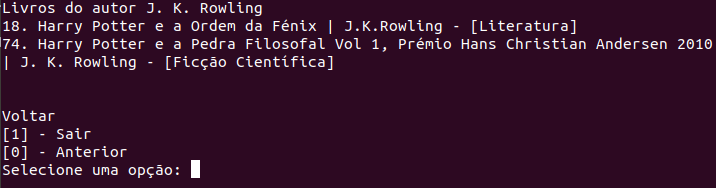
12. Screenshot da possiblidade de existência de um autor



13. Screenshot da não existência de um autor

4.5.3.2 Listar livros do autor

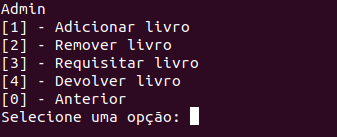
Se no opção do tópico anterior **4.5.3.1 Verificar se há livros do autor** existir a possibilidade de, realmente, o autor recebido do utilizador ter livros seus na biblioteca, então é usada a *Class* **MinHash**, e verificação de similaridade, à semelhança do que é feito nos tópicos **4.5.2.2 Listar livros com títulos parecidos** e **4.5.1.2 Listar livros por categoria**, para filtrar os livros desse autor e listá-los na consola.



14. Screenshot da listagem de livros do autor

**4.5.4 Administração**

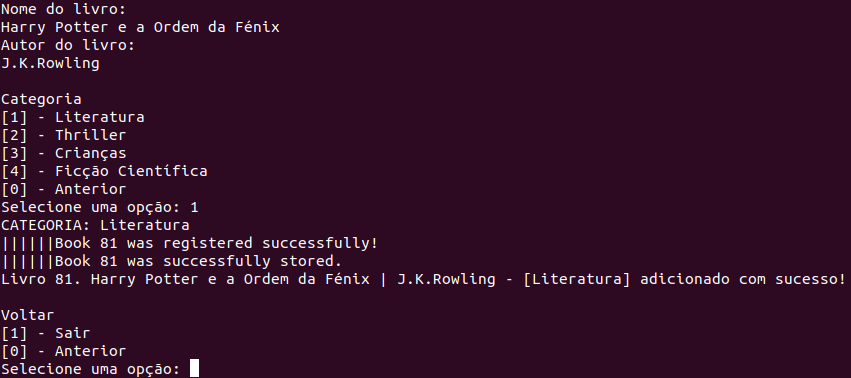
Nesta secção, o utilizador tem acesso a quatro opções: Adicionar livro, Remover Livro, Requesitar livro e Devolver livro. Estas opções trabalham diretamente na edição do objeto da *Class* **Library** e dos objetos da *Class* **Book**.



15. Screenshot do menu da Administração

4.5.4.1 Adicionar livro

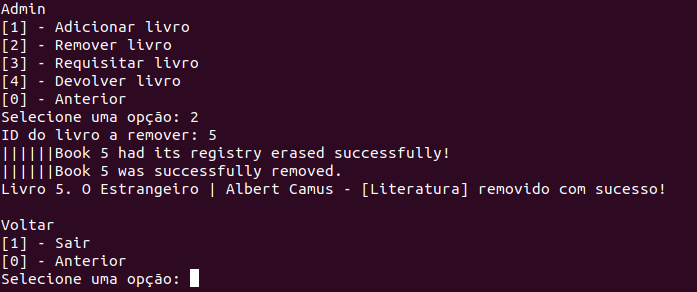
Esta opção recebe do utilizador as informações do livro que quer adicionar, como o título e o autor, e dá a escolher ao utilizador qual a categoria que este se enquadra. É usada a função **Book.addBook()**. O ID do livro é obtido automaticamente através da informação do ID do último livro do acervo.



16. Screenshot da adição de um livro com sucesso

4.5.4.2 Remover livro

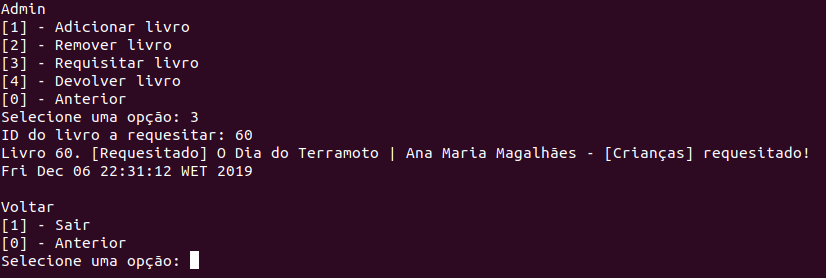
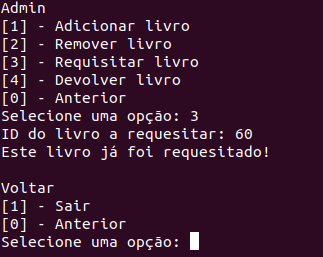
Esta opção recebe do utilizador o ID do livro que quer elimar. É usada a função **Book.removeBook()**.



17. Screenshot da remoção de um livro com sucesso

4.5.4.3 Requesitar livro

Esta opção recebe do utilizador o ID do livro que quer requesitar. É usada a função **Book.borrowed()** que verifica se o livro já foi, ou não requesitado, e caso esteja disponível para requesitar, é usada a função **Book.borrow()**. É também guardada e displayed a data de requesito do livro.



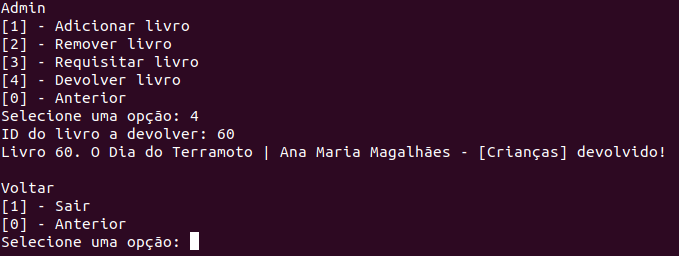
19. Screenshot de uma tentativa de requesito de um livro já requesitado

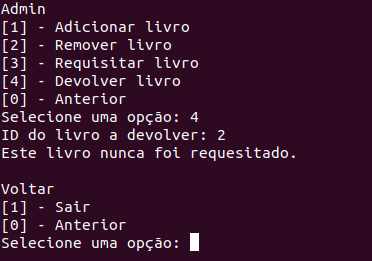
18. Screenshot do requesitar de um livro com sucesso

4.5.4.3 Devolver livro

Esta opção recebe do utilizador o ID do livro que quer devolver. É usada a função **Book.borrowed()** que verifica se o livro já foi, ou não requesitado, e caso esteja tenha sido realmente requesitado, é usada a função **Book.returnBook()**.

­­



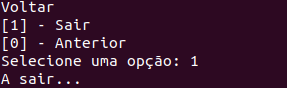


21. Screenshot de uma tentativa de devolução de um livro não requesitado

20. Screenshot da devolução de um livro com sucesso

**4.5.5 Abandono da ação**

Existem menus que aparecem com alguma frequência, que possibilitam o utilizador de sair do programa, ou simplesmente voltar para o menu principal.



22. Screenshot do menu de retrocesso