

Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Ciência da Computação

Trabalho Prático de Programação Orientada a Objetos

BCC221 - Programação orientada a objeto

Felipe Marques, Lucas Chagas, Matheus Peixoto, Nicolas Mendes, Pedro Henrique de
Oliveira, Pedro Moraes

Professor: Guillermo Camara Chavez

Ouro Preto
12 de agosto de 2023

Sumário

1	Introdução	1
1.1	Objetivo Geral	1
1.2	Objetivos Específicos	1
1.3	Justificativa	1
1.4	Especificações técnicas do projeto	1
1.5	Especificações da máquina de teste	1
2	Desenvolvimento	1
2.1	Diagrama das classes do pacote Modelo	2
2.1.1	Classe Pessoa	3
2.1.2	Classe Funcionário	3
2.1.3	Classe Usuário	3
2.1.4	Classe Autor	3
2.1.5	Classe Categoria	3
2.1.6	Classe Livro	3
2.1.7	Classe Empréstimo	4
2.2	Diagrama das classes do pacote Dados	4
2.2.1	Classe Dados	5
2.3	Pacote DAO e a interface DAOInterface	5
2.3.1	Interface DAOInterface	5
2.3.2	Exemplo de classe do pacote DAO que implementa a interface DAOInterface	6
2.3.3	Observações relevantes acerca da classe DAOLivro	6
2.4	TableModel do pacote Tabelas	7
2.5	Telas da aplicação	7
3	Utilização do programa	17
3.1	Usuário	17
3.2	Funcionário	17
3.3	Administrador	17
4	Conclusão	18

Lista de Figuras

1	Diagrama UML	2
2	Diagrama dados	4
3	UML das classes do pacote DAO	5
4	Menu principal - Login A primeira tela que se tem ao executar o programa é o menu principal . Através do mesmo é possível logar como administrador, funcionário ou usuário a partir do id, sendo o único id do administrador "123456"	8
5	Menu - Administrador Entrando no sistema como administrador, temos um menu com 6 opções: "Funcionários", "Usuários", "Autores", "Categorias", "Livros" e "Sair", onde as 5 primeiras opções levam para uma tela na qual permite o cadastro, a edição e a remoção de cada item respectivo, e a opção sair retorna para tela de login.	9
6	Tela autores - Administrador Esta tela permite ao administrador inserir, editar e remover autores, além de visualizar uma tabela com os autores cadastrados.	9
7	Tela categorias - Administrador Esta tela permite ao administrador inserir, editar e remover categorias, além de visualizar uma tabela com as categorias cadastradas.	10

8	Tela de livros - Administrador	
	Esta tela permite ao administrador inserir, editar e remover livros, além de visualizar uma tabela com os livros cadastrados. Ao clicar em um livro na tabela é possível visualizar e manipular os autores e as categorias referentes do mesmo.	10
9	Tela de edição de autor(es) de um livro - Administrador	
	Esta tela permite ao administrador adicionar e remover autores, cadastrados previamente no sistema, vinculados a um livro específico.	11
10	Tela de edição de categoria(s) de um livro - Administrador	
	Esta tela permite ao administrador adicionar e remover categorias, presentes no sistema, vinculadas a um livro específico.	11
11	Tela de funcionário - Administrador	
	Esta tela permite ao administrador inserir, editar e remover funcionários, além de visualizar uma tabela com os funcionários cadastrados.	12
12	Tela de usuário - Administrador	
	Esta tela permite ao administrador inserir, editar e remover usuários, além de visualizar uma tabela com os usuários cadastrados.	12
13	Menu funcionário	
	Ao entrar no sistema como funcionários temos 2 opções, uma para cadastrar empréstimos de livros, além de editar e remover os mesmos, e outra opção para sair do sistema. . . .	13
14	Tela de empréstimo - Funcionário	
	Esta tela permite ao funcionário cadastrar, editar e remover empréstimos de livros utilizando os ids do usuário e do livros e a data referentes a este empréstimo. É possível selecionar o usuário e o livro apenas clicando nos itens desejados nas tabelas de usuários e livros disponíveis na parte inferior da tela.	14
15	Tela do Acervo da biblioteca - Usuário	
	Esta tela permite ao usuário consultar tanto os livros cadastrados no sistema, quanto os empréstimos de livros referentes a ele. Foi utilizado o componente JTabbedPane para que fosse possível o usuário transitar entre as duas tabelas na mesma tela de forma organizada.	15
16	Tela dos Empréstimos vinculados - Usuário	
	Esta tela permite ao usuário visualizar todos os empréstimos de livros, registrados no sistema, vinculados ao mesmo	16

1 Introdução

Neste trabalho prático, os conceitos de Programação Orientado a Objetos, passados em sala de aula, na linguagem Java, serão utilizados para a composição de um projeto que atenda às demandas presentes no documento orientador.

1.1 Objetivo Geral

O objetivo geral deste trabalho prático é o desenvolvimento de uma aplicação em Java, com interface gráfica, que atue como um sistema de gerenciamento de empréstimos de livros em uma biblioteca.

1.2 Objetivos Específicos

Este trabalho possui como objetivos específicos:

- Implementar um sistema de cadastro de funcionários, livros, usuários, empréstimos, autores e categorias, no qual o administrador não precisa estar cadastrado no sistema.
- Concretizar um sistema de empréstimo de livro, que viabilize que um funcionário no sistema realize a operação de empréstimo a um usuário comum.
- Viabilizar que o usuário efetue consultas aos livros cadastrados no sistema.
- O programa deverá manipular os dados em caráter temporário, excluindo a permanência dos dados após a finalização da aplicação.

1.3 Justificativa

A fim de apresentar, de modo prático, por meio da implementação do programa requisitado, na linguagem Java, os conceitos de Programação Orientada a Objetos passados em sala de aula, este trabalho prático serve para assimilação e aprimoramento dos mesmos.

1.4 Especificações técnicas do projeto

- IDE utilizada: Apache NetBeans
- Versão da IDE: 17
- Versão do JDK (Java Development Kit): 17

1.5 Especificações da máquina de teste

- Intel Core i5 9ª geração
- 16 GB de RAM
- SSD NVMe
- WSL utilizando o Ubuntu 22.04.2 LTS

2 Desenvolvimento

A figura representa o diagrama UML que foi utilizado para a realização do trabalho. Assim, os próximos tópicos abordarão as classes, começando por aquelas que funcionam sem necessitar de outras, para as que possuem um pouco mais e terminando com as que de fato necessitam das outras.

2.1 Diagrama das classes do pacote Modelo

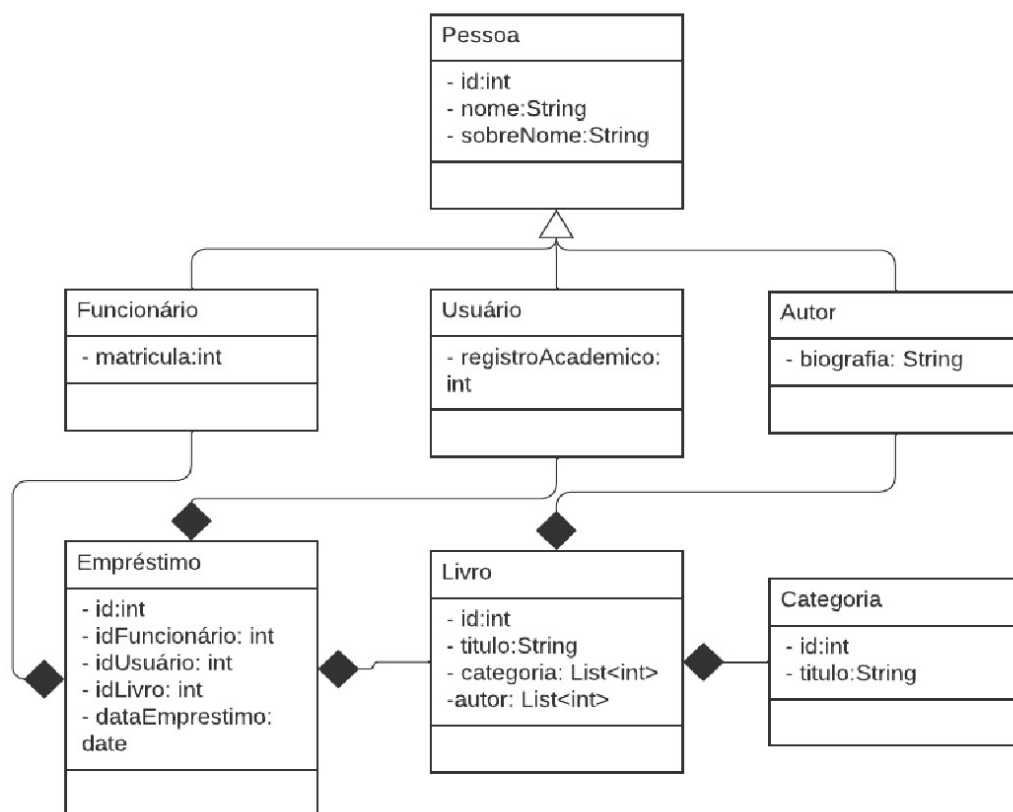


Figura 1: Diagrama UML

2.1.1 Classe Pessoa

A classe Pessoa possui três atributos, sendo eles um id, do tipo inteiro, que representa um campo numérico único identificador, um nome, do tipo String, que representa o primeiro nome da pessoa e, um sobrenome, que representa o restante do nome da pessoa e, por fim, há um inteiro estático, que representa um contador que será o ID incrementado automaticamente a cada criação de Pessoa. Em síntese, essa classe possui os atributos básicos e mutualmente compartilhados pelas classes que herdam de Pessoa. Ademais, possui a implementação de getters e setters para cada atributo, o método equals, o método toString e dois construtores, um em que o ID atribuído ao novo objeto é recebido via parâmetro e o outro o ID recebe o valor do contador auto incrementado, os demais atributos são recebidos por parâmetros e setados em ambos construtores.

2.1.2 Classe Funcionário

A classe Funcionário herda da classe Pessoa e possui o atributo matricula, do tipo inteiro, que representa o número de matrícula do funcionário. Outrossim, a classe possui getter e setter para o atributo matricula, a implementação do método atualizaFuncionario(), no qual recebe um objeto da classe Funcionario e utiliza os métodos setters da classe Base e o próprio setter da classe para a atualização do Funcionário, e a implementação do método equals. Por fim, a classe Funcionário, por meio de uma relação de composição, constituirá um atributo da classe Empréstimo.

2.1.3 Classe Usuário

A classe Usuário herda da classe Pessoa e possui o atributo regAcademico, do tipo inteiro, que representa o registro acadêmico do usuário cadastrado. Ainda, a classe possui getter e setter para o atributo regAcademico, a implementação de dois construtores, com um deles possuindo o atributo id, a elaboração de um método atualizaUsuario(), que recebe um objeto da classe Usuario e utiliza os setters da classe Base e o próprio setter da classe em questão para a efetivação desta função, além da implementação do método equals. Por fim, a classe Usuário, por meio de uma relação de composição, constituirá um atributo da classe Empréstimo.

2.1.4 Classe Autor

A Classe Autor herda da classe Pessoa e possui apenas um atributo biografia que é um objeto da classe String. Os métodos desta classe são os getters e setters (para o único atributo da classe), equals, os dois construtores, sendo que um deles constitui o atributo id, o método atualizaAutor(), que recebe um objeto da classe Autor e utiliza os setters da classe Base para tal atualização e também o setter da classe em questão, e o toString. Sua utilidade consiste em fazer parte da classe "Livro" como uma composição.

2.1.5 Classe Categoria

A classe Categoria é composta por três atributos, sendo um atributo do tipo static int, no qual é um contador que será incrementado na variável identificador no construtor, que é um atributo do tipo inteiro. Ademais, a classe é constituída também por um atributo do tipo String, que será o título da categoria. A vista disso, estão implementados os getters e setters da classe, um método toString, o método atualizaCategoria(), que recebe um objeto da classe Categoria e utiliza o setter da classe para a atualização do título, e também o método equals para a elaboração de comparações. Por fim, a classe Categoria, por meio de uma relação de composição, constituirá um atributo da classe Livro.

2.1.6 Classe Livro

A Classe Livro possui os seguintes atributos: um 'id' representado por um inteiro, um "título" representado por uma String, uma sessão de categoria representado por uma lista e uma lista de autores. Os métodos da classe consistem em getters e setters para o 'id' e para o

'titulo', os métodos adicionar() e remover() para autores e categorias, o método atualizaLivro(), que recebe um objeto da classe Livro e utiliza o setter do titulo para mudá-lo, e os construtores. Por fim, a classe Livro, por meio de uma relação de composição, constituirá um atributo da classe Empréstimo.

2.1.7 Classe Empréstimo

A classe Empréstimo possui cinco atribuídos, sendo que quatro atributos de tal classe são inteiros, que representam identificadores do empréstimo, do funcionário, do usuário e do livro e o ultimo atributo do tipo Date, no qual representa a data do dia do empréstimo. Através disso, e implementado os métodos de getters e setters, os dois construtores, o metodo atualizaEmprestimo(), no qual recebe um objeto da classe Empréstimo e através da utilização de setters da classe em questão, o empréstimo é atualizado e também o método equals.

Vale ressaltar que em todas as classes, exceto a Pessoa, temos um método atualiza, que recebe um objeto do tipo da classe e altera os valores de todos os atributos das mesmas, exceto o ID.

Este método foi criado para que facilitasse a atualização dos mesmos, uma vez que a ideia inicial era gerar um novo objeto e alterar o mesmo no vetor da classe Dados. Todavia isso acaba gerando problemas durante a alteração de dados em classes que agregam outras classes. Supondo como exemplo um objeto do tipo Livro, o mesmo possui um vetor de categoria, onde os dados são obtidos a partir do vetor da classe Dados, porém, quando uma categoria era alterada, a modificação não refletia no objeto Livro. Assim, métodos como o clone para gerar uma cópia do objeto ficariam inviáveis.

2.2 Diagrama das classes do pacote Dados

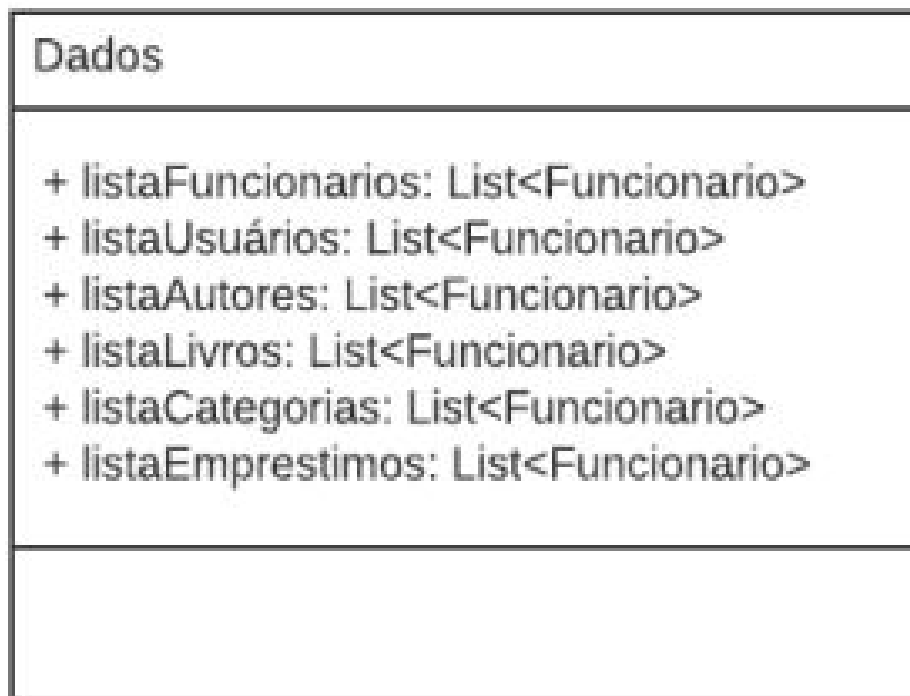


Figura 2: Diagrama dados

2.2.1 Classe Dados

A classe Dados é formada por seis atributos públicos e estáticos do tipo List, sendo cada uma das variáveis para armazenar dados (objetos) das classes Funcionário, Usuário, Autor, Livro, Categoria e Empréstimo. Tal classe é implementada para representar um banco de dados do sistema e assim, ser usada para o funcionamento do sistema de gerenciamento de empréstimo, em que as operações com os dados são efetuadas através dos métodos das classes do pacote DAO, que implementam a interface DAO para manipulação dos objetos das classes do pacote Modelo, assegurando a interação com o banco de dados e métodos primários das classes principais.

Vale ressaltar que, mesmo os dados sendo públicos, eles não são acessados de fora direta por nenhuma parte do programa além das DAO's. Sendo que esta medida foi tomada a fim de representar mais fielmente um um banco de dados, onde não é possível alterar de forma direta os registros.

2.3 Pacote DAO e a interface DAOInterface

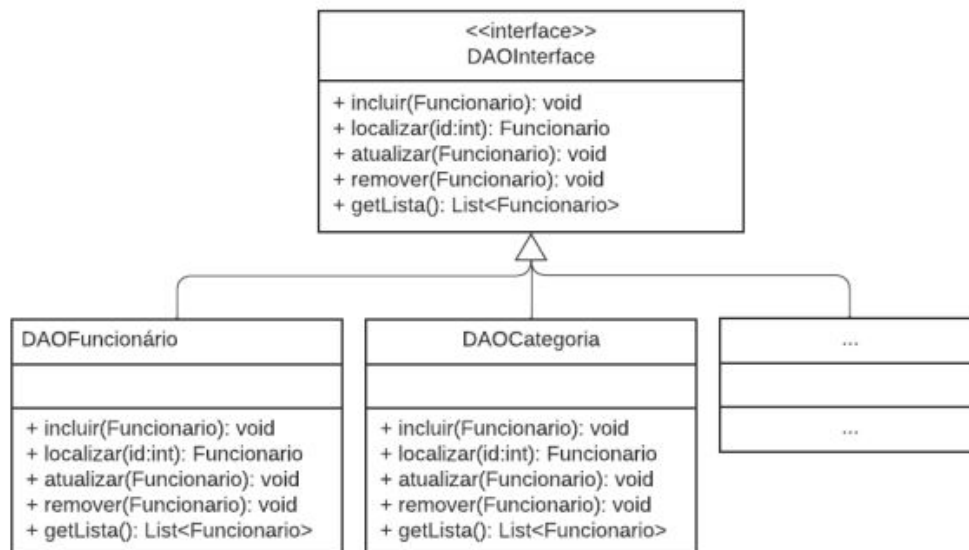


Figura 3: UML das classes do pacote DAO

2.3.1 Interface DAOInterface

A interface DAOInterface possui o protótipo de 5 métodos que serão implementados por todas as classes pertencentes ao pacote DAO. Tais métodos correspondem a manipulação de objetos instanciados, conforme estabelecido pelo padrão DAO (Data Access Object), das classes referentes ao pacote Modelo, portanto, a interface DAOInterface será implementada por 6 classes, com exceção da classe Pessoa, pois, durante a execução do programa não serão instanciados objetos desta classe, que é abstrata.

Com relação a função dos métodos, temos que: o método incluir() é responsável por fazer a inserção do objeto na lista; o método localizar() que, dado um ID passado via argumento, retorna, caso exista, o objeto com o ID informado presente na lista de objetos; o método atualizar() que recebe dois objetos via argumento, um que corresponde ao objeto com os novos dados e outro com os valores antigos, e, após verificar a validade da atualização, busca o objeto a ser alterado na lista de objetos e, caso exista, o objeto original receberá os valores do novo a partir do método atualizar de sua respectiva classe; o método remover() que recebe um objeto

via argumento e o remove da lista de objetos, caso exista e o método `getLista()` que retorna uma lista de objetos proveniente da classe `Dados`.

Ademais, o construtor da classe inicializa todas as listas.

2.3.2 Exemplo de classe do pacote DAO que implementa a interface DAOInterface

A fim de tornar o relatório o mais sintético e objetivo possível, a classe `DAOAutor` exemplificará a implementação dos métodos da interface `DAOInterface`, que seguem a mesma estrutura e lógica para as demais classes.

A classe `DAOAutor` implementa todos os métodos da interface `DAOInterface` e, com relação aos métodos, temos:

O método `incluir()` recebe um `Object`, realiza o cast para a classe correspondente, nesse caso, a classe `Autor`, e, por meio do método `add()`, adiciona o objeto a lista de objetos da classe correspondente, nesse caso, a lista de autores.

O método `localizar()` recebe um inteiro, via parâmetro, representando um ID, e, por meio de um for convencional, ele varre a lista de autores, comparando os IDs com o ID passado via argumento, e, caso seja encontrado o autor com o ID informado, o mesmo é retornado. Caso todo o vetor seja percorrido, isso significa que o item não existe no vetor, então é retornado um objeto nulo.

O método `atualizar()` recebe dois objetos, um a ser alterado e outro contendo as alterações, realiza o cast de ambos para objetos da classe `Autor`, e, seguidamente, verifica se os objetos são iguais, através do método `equals()`, e se algum dos objetos estão apontando para null, a fim de que o processo de atualização não realize operações desnecessárias. Posteriormente, busca-se o índice do autor a ser alterado na lista de autores e, caso exista, chamamos o método `atualizar` do autor original e passamos como argumento o objeto com os novos dados.

O método `remover()` recebe um objeto e, seguidamente, verifica se o objeto está apontando para null, a fim de garantir que a remoção ocorra levando em consideração um objeto válido. Na sequência, realiza-se um cast para um objeto da classe `Autor` e, por meio do método `remove()` para `List`, o autor, caso esteja presente no `List`, será efetivamente removido da lista de autores.

O método `getLista()` retorna a lista de objetos da classe correspondente, nesse caso, uma lista de autores.

2.3.3 Observações relevantes acerca da classe DAOLivro

A classe `DAOLivro` apresenta alguns métodos a mais do que as demais classes do pacote DAO, pois, em função das especificações dos requisitos do trabalho, a classe `Livro` possui objetos de outras classes como atributos dela mesma (classes `Categoria` e `Autor`). Portanto, em função da natureza da construção da classe supracitada, alguns métodos a mais foram implementados, com o fito de abordar tratativas coerentes para a questão referida. Sendo que os mesmos foram criados para facilitar o processo de inserção de categorias e autores através da interface gráfica que usará um `TableModel` para exibir os itens.

Dentre esses métodos, temos:

O método `adicionarCategoria()`, que recebe um inteiro, representando o ID do livro desejado, e um objeto da classe `Categoria`. Seguidamente, por meio do método `localizar()` e com o ID passado via argumento, o livro com o ID desejado é localizado dentro da lista de livros, caso exista, e por meio do método `adicionarCategoria()`[da classe `Livro`], a categoria é adicionada à lista de categorias do livro correspondente.

O método `adicionarAutor()`, que recebe um inteiro, representando o ID do livro desejado, e um objeto da classe `Autor`. Continuamente, o livro, caso exista na lista de livros, é encontrado pelo método `localizar()` e o ID recebido, e, na sequência, o autor é adicionado à lista de autores do referido livro, através do método `adicionarAutor()`[da classe `Livro`].

O método `removerAutor()`, que recebe um inteiro, representando o ID do livro desejado, e um objeto da classe `Autor`. Sucessivamente, o livro, caso esteja presente na lista de livros, é localizado, por meio do método `localizar()` e o ID recebido, e o autor é removido da lista de autores do livro em questão, por meio do método `removerAutor()`[da classe `Livro`].

O método `removerCategoria()`, que recebe um inteiro, representando o ID do livro desejado, e um objeto da classe `Categoria`. Na sequência, o livro com o ID desejado é localizado na lista de livros, por meio do método `localizar()` e do ID recebido, caso exista, e, através do método `removerCategoria()` [da classe `Livro`], a categoria é excluída da lista de categorias do livro referido.

2.4 TableModel do pacote Tabelas

Todos os dados são exibidos em tabelas, dessa forma, para facilitar o uso das mesmas, foram criados `TableModel`'s para todas as classes do pacote `Modelo`, sendo que todas as classes seguem a mesma estrutura de implementar os métodos `getRowCount()`, `getColumnCount()`, `GetColumnName()` e `getValueAt()` da interface `AbstractTableModel`. Ademais, todos possuem um vetor que representa os seus dados, que são os valores que estão na classe `Dados`.

Os modelos possuem um ou dois construtores, sendo um sem parâmetro e o outro com, sendo que aquele que não recebe dados está atribuindo a referência do vetor da classe para o respectivo vetor da classe `Dados`, para que qualquer modificação no vetor reflita na tabela. Ademais, todos os modelos apresentam um método para adicionar itens na tabela, atualizar e deletar, que são chamados pelas telas do sistema, sendo que estas funções chamam os métodos corretos de cada DAO e depois atualiza a tabela com o `fireTableDataChanged()`, pois como o vetor que a classe referencia foi mudado, então há novas modificações para serem feitas na tabela.

O segundo construtor de **TabelaAutor** recebe um vetor de `Autor` como argumento e é utilizado na tela de cadastro de livros, mais especificamente no menu de autores, onde, com um livro selecionado, exibimos nessa tabela somente os livros atribuídos ao livro correspondente.

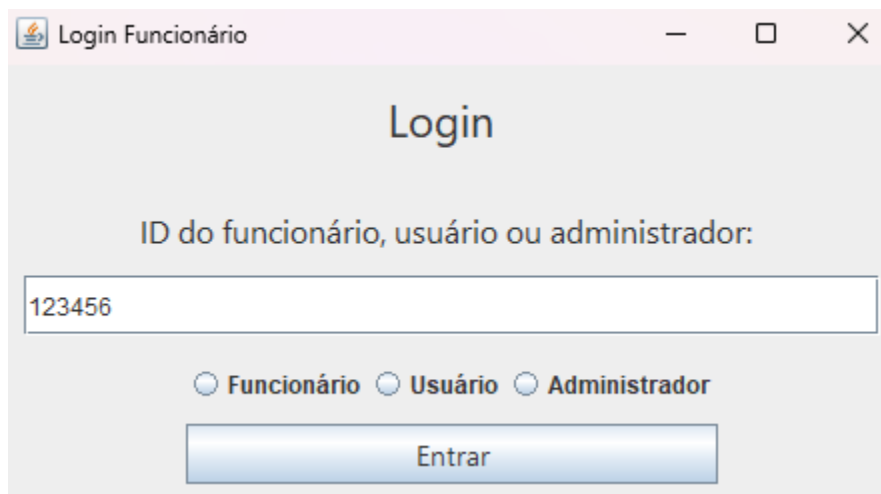
A classe **TableCategoria** segue o mesmo princípio e funcionamento que a classe explicada anteriormente.

É importante ressaltar que estas classes possuem métodos que as outras não possuem, sendo eles `addCategoriaLivro()` e `deletarCategoriaLivro()` para a classe `TabelaCategoria`, e `addAutorLivro()` e `deletarAutorLivro()`. Essas funções fazem-se necessárias uma vez que, quando estamos alterando os autores e categorias dos livros, estamos lidando diretamente com as tabelas, assim, o processo de adicionar itens e atualizar a tabela fica simplificado. já que, quando uma alteração é necessária, estes métodos são chamados passando uma categoria, ou autor, e o livro onde as mudanças ocorrerão. Em seguida, as verificações necessárias são feitas, o DAO correto é chamado e o comando de alterar a tabela é executado sem grandes complicações. Dessa forma, como estamos alterando uma lista de dados que está sendo exibida na tela, fica coerente a chamada para a sua modificação ser no modelo de tabela respectivo.

A classe **TabelaEmprestimo** conta somente com um construtor a mais, sendo que o mesmo recebe um usuário e procura na lista de empréstimos todos aqueles que possuem o id do usuário em questão. Dessa forma, o vetor da classe só terá alguns itens, pois, na tela de menu do usuário, ele somente precisa verificar os seus empréstimos e não o dos outros usuários.

Já as classes **TabelaFuncionario**, **TabelaUsuario** e **TabelaLivro** possuem as mesmas funções, variando somente o tipo de dados. Todavia o modelo para a tabela de livros possui uma pequena diferença, onde no método `getValueAt`, ao invés de simplesmente retornar um dados, é feito uma varredura no vetor de autores e categorias para que a exibição dos dados seja feito de uma forma mais apresentável.

2.5 Telas da aplicação



Login

ID do funcionário, usuário ou administrador:

123456

☒ Funcionário ☐ Usuário ☐ Administrador

Entrar

Figura 4: Menu principal - Login

A primeira tela que se tem ao executar o programa é o **menu principal**. Através do mesmo é possível logar como administrador, funcionário ou usuário a partir do id, sendo o **único id do administrador "123456"**.



Figura 5: **Menu - Administrador**

Entrando no sistema como administrador, temos um menu com 6 opções: "Funcionários", "Usuários", "Autores", "Categorias", "Livros" e "Sair", onde as 5 primeiras opções levam para uma tela na qual permite o cadastro, a edição e a remoção de cada item respectivo, e a opção sair retorna para tela de login.

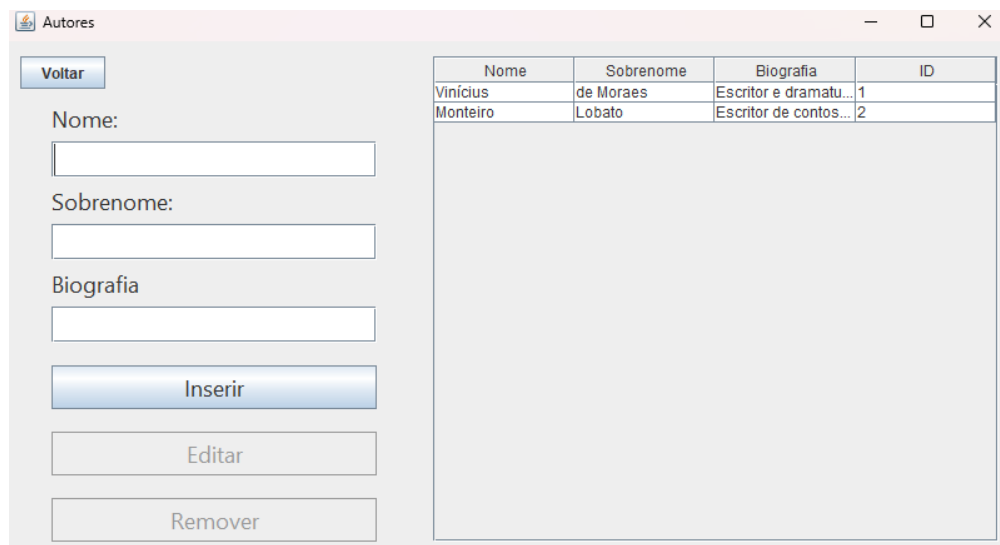


Figura 6: **Tela autores - Administrador**

Esta tela permite ao administrador inserir, editar e remover autores, além de visualizar uma tabela com os autores cadastrados.

Nome	id
Romance	1
Ação	2
Filosofia	3
História em quadrinhos	4

Figura 7: **Tela categorias - Administrador**

Esta tela permite ao administrador inserir, editar e remover categorias, além de visualizar uma tabela com as categorias cadastradas.

Título	Autor(es)	Categoria(s)	id
Deuses americanos		Romance, Ação	1
Capitães de areia		Filosofia, Roma...	2

Figura 8: **Tela de livros - Administrador**

Esta tela permite ao administrador inserir, editar e remover livros, além de visualizar uma tabela com os livros cadastrados. Ao clicar em um livro na tabela é possível visualizar e manipular os autores e as categorias referentes do mesmo.



Figura 9: **Tela de edição de autor(es) de um livro - Administrador**

Esta tela permite ao administrador adicionar e remover autores, cadastrados previamente no sistema, vinculados a um livro específico.

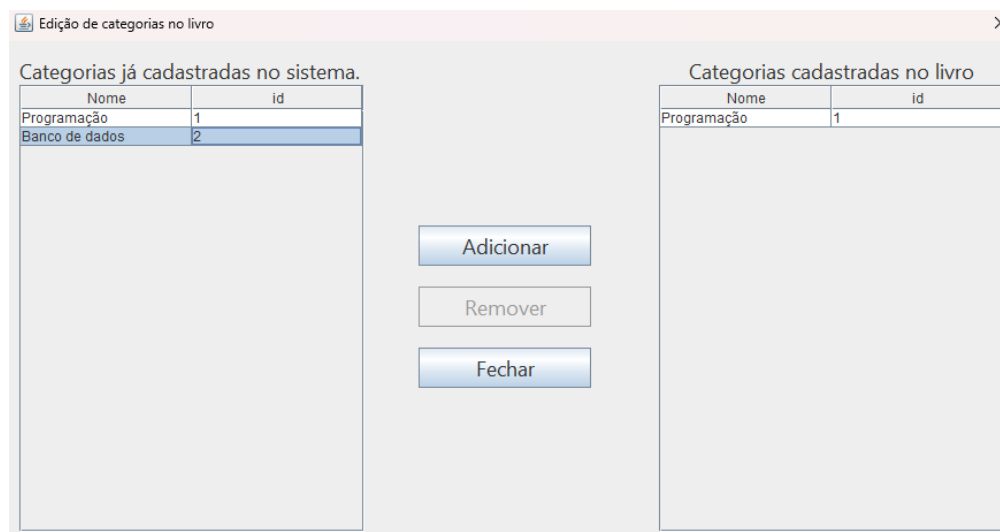


Figura 10: **Tela de edição de categoria(s) de um livro - Administrador**

Esta tela permite ao administrador adicionar e remover categorias, presentes no sistema, vinculadas a um livro específico.

Nome	Sobrenome	Matrícula	ID
Lucas	Peixoto	57845	5
Pedro	Marques	69697	6

Figura 11: **Tela de funcionário - Administrador**

Esta tela permite ao administrador inserir, editar e remover funcionários, além de visualizar uma tabela com os funcionários cadastrados.

Nome	Sobrenome	Reg. Acadêmico	ID
Nicolas	Moreira	50957	7
Matheus	Oliveira	45512	9

Figura 12: **Tela de usuário - Administrador**

Esta tela permite ao administrador inserir, editar e remover usuários, além de visualizar uma tabela com os usuários cadastrados.

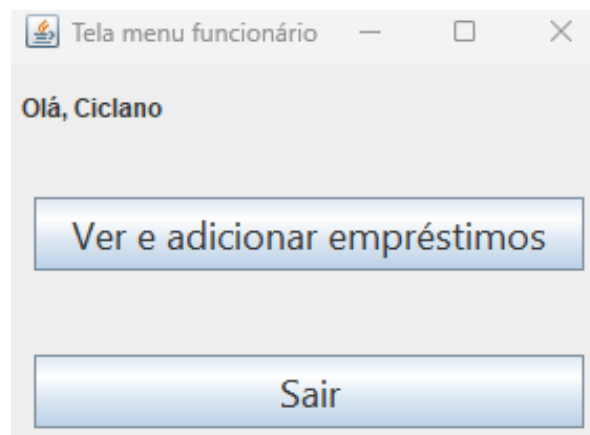


Figura 13: **Menu funcionário**

Ao entrar no sistema como funcionários temos 2 opções, uma para cadastrar empréstimos de livros, além de editar e remover os mesmos, e outra opção para sair do sistema.

Empréstimos

Voltar

ID do usuário:

ID do Livro:

Data do Empréstimo:

//

Hoje

Inserir

Editar

Remover

Empréstimos

Funcionário	Usuário	Livro	Data	ID
-------------	---------	-------	------	----

Usuários Disponíveis

Nome	Sobrenome	Reg. Acadêmico	ID
Pedro	Leão	123	5
Felipe	Braz	546	6

Livros Disponíveis

Título	Autor(es)	Categoria(s)	id
POO	Alan Kay	Programação	1
Programação	Alan Turing	Programação	2

Figura 14: **Tela de empréstimo - Funcionário**

Esta tela permite ao funcionário cadastrar, editar e remover empréstimos de livros utilizando os ids do usuário e do livros e a data referentes a este empréstimo. É possível selecionar o usuário e o livro apenas clicando nos itens desejados nas tabelas de usuários e livros disponíveis na parte inferior da tela.

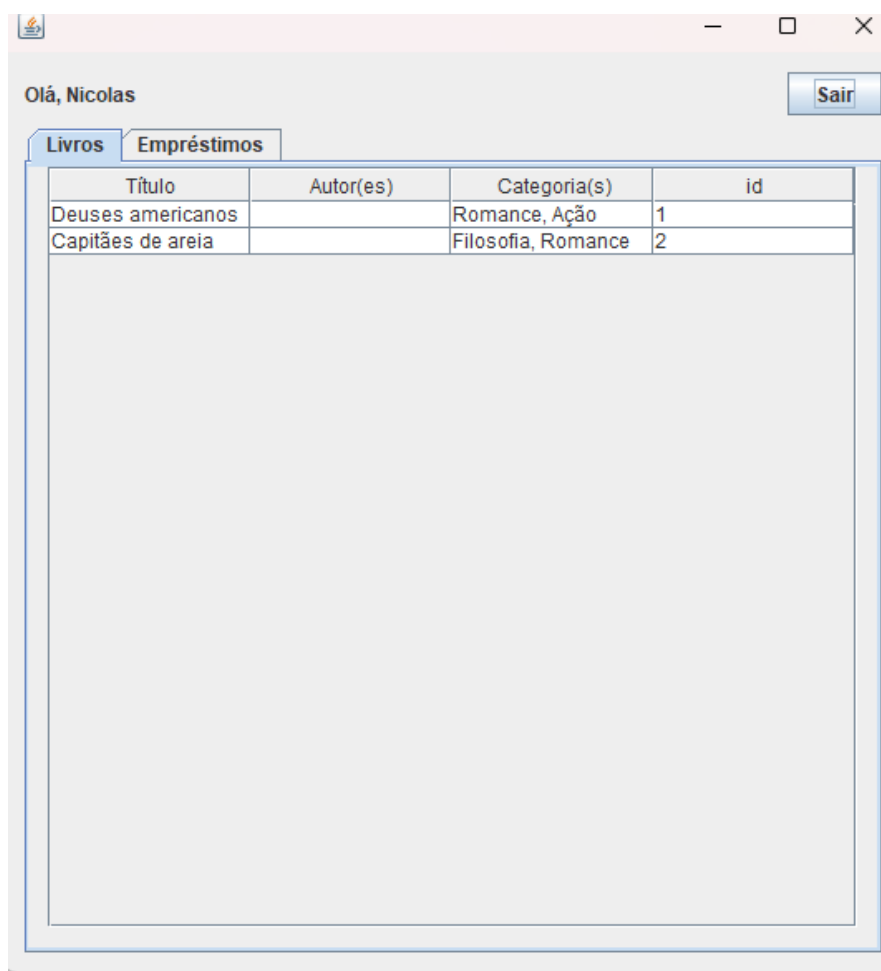


Figura 15: **Tela do Acervo da biblioteca - Usuário**

Esta tela permite ao usuário consultar tanto os livros cadastrados no sistema, quanto os empréstimos de livros referentes a ele. Foi utilizado o componente `JTabbedPane` para que fosse possível o usuário transitar entre as duas tabelas na mesma tela de forma organizada.

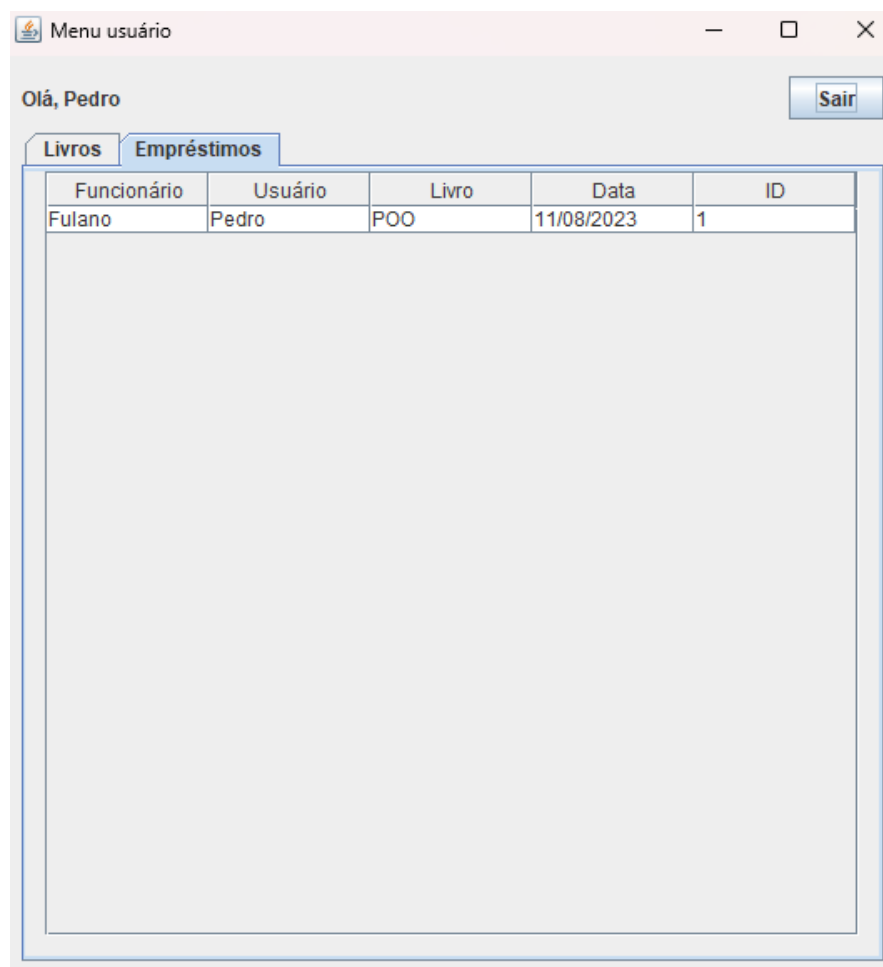


Figura 16: Tela dos Empréstimos vinculados - Usuário

Esta tela permite ao usuário visualizar todos os empréstimos de livros, registrados no sistema, vinculados ao mesmo

3 Utilização do programa

Ao iniciar o programa, o usuário irá depara-se com uma tela de Login, onde ele deverá adicionar o ID de um funcionário, usuário ou a sequência 123456 para acessar o campo de administrador e marcar qual o tipo de usuário ele é. Caso o ID não seja encontrado para o tipo de acesso solicitado, uma mensagem de erro é exibida.

Vale ressaltar que a fim de testes, foram criados, no arquivo "TrabalhoPratico", valores já pré-definidos para o login como usuário e funcionário. Dessa forma, pode-se usar os ID's 3 e 4 para funcionário e 5 e 6 para usuário.

3.1 Usuário

A tela de menu do usuário conta com duas janelas possíveis de serem acessadas, sendo a primeira uma lista com todos os livros disponíveis e a segunda uma lista com todos os empréstimos que ele realizou.

3.2 Funcionário

O funcionário pode somente cadastrar empréstimos, dessa forma, para realizar tal ação, no menu de cadastro, listagem e edição, o ID do usuário que realizará o empréstimo pode ser adicionado tanto por escrita quanto selecionando na lista de "Usuários Disponíveis", sendo que o mesmo vale para o ID do livro.

Para editar um empréstimo, basta clicar no item desejado na tabela "Empréstimo" que todos os seus valores irão para o formulário. Assim, pode-se mudar os valores, inserir um novo empréstimo, salvar as alterações clicando em "Editar" ou deletar o registro clicando em "Remover".

3.3 Administrador

Na tela de administrador há os menus de Funcionários, Usuários, Autores, Categorias e Livros, sendo cada um deles explorados a seguir:

Nos menus de "funcionário", "Usuário", "Autor" e "Categoria", que possuem comportamentos similares, podemos fazer os respectivos cadastros a partir do correto preenchimento do formulário.

Após inserido, clicando no cadastro desejado, na tabela, os seus dados irão voltar ao formulário, e lá eles podem ser modificados e inseridos novamente na tabela como forma de um novo registro pelo botão "Inserir", modificado e salvo pelo botão de "Editar" ou simplesmente apagado pelo botão de "Remover".

Já a tela "Livros", cadastramos um novo valor inserindo apenas o título e clicando em "Inserir". Em seguida, podemos editar e remover um livro da mesma forma que nas outras telas. Todavia, quando um livro é selecionado, podemos adicionar e remover autores e categorias clicando nos botões "Autores" e "Categorias", respectivamente, onde os dois possuem o mesmo comportamento.

Assim, para a modificação desses registros, é necessário selecionar um item na tabela da esquerda e clicar em "Adicionar" para que o valor indicado seja devidamente inserido no objeto do livro, sendo que, para remover, basta clicar no item desejado na tabela da direita e, em seguida, clicar em "Remover". As alterações são salvas no mesmo instante, assim, a tela pode ser fechada no momento em que for necessário.

Por fim, vale ressaltar que qualquer edição na categoria ou no autor será refletida nos respectivos itens dos livros, sendo que o mesmo vale para quando um livro for modificado, logo, a tabela de empréstimos também será modificada.

4 Conclusão

Com o fim do trabalho, foi possível compreender mais sobre o desenvolvimento de programas utilizando o paradigma de programação orientada a objetos, implementando todos os conceitos, aprendidos anteriormente na linguagem C++, na linguagem Java.

Para a execução deste trabalho prático, foram utilizadas as propriedades e concepções básicas, aprendidas em sala de aula, como herança, composição, agregação, polimorfismo, encapsulamento, lançamento e tratamento de exceções, empreendimento do protocolo DAO (por meio de classes para manipulação de objetos), e projeção e integração de interface gráficas.

Com relação às principais problemáticas vivenciadas pelo grupo, no geral, um dos maiores problemas que o grupo teve foi em relação a como trabalhar com a classe Date, pois tivemos problema com a formatação do date. Outro fator um pouco complicado foi o entendimento das tabelas, e a sua logística de atualização, pois a biblioteca que foi trabalhada em sala de aula (Binding beans), não existe mais nas versões mais recentes do NetBeans, logo tivemos que aprender por fora novas formas para manipular as tabelas, através da construção de um table model próprio.

Ademais, outro problema recorrentemente experienciado pelo grupo foi a questão de perda de referência de objetos, uma vez que, na linguagem Java, essa questão do que é alocado estaticamente ou dinamicamente fica a cargo da própria linguagem, diferentemente de C++, em que a manipulação de ponteiro viabiliza a constituição, de modo explícito, de uma estratégia altamente eficiente e controlável, por parte do programador, para essa gestão de referência e memória.]

Entretanto, com o auxílio do material didático disponibilizado na plataforma Moodle, juntamente, com algumas pesquisas na internet, todos esses problemas foram devidamente contornados e solucionados, fazendo com que todos os requisitos e demandas do trabalho fossem devidamente atendidos, viabilizando a implementação idealizada do trabalho.

Em suma, o propósito enunciado na introdução foi integralmente satisfeito durante o processo de empreendimento do TP, permitindo aos membros do grupo compreender a aplicação, com viés prático e objetivo, de todo conhecimento adquirido durante as aulas da disciplina. Ademais, um fato interessante a citar