

# **Desenvolvimento de sistema para manipulação de biblioteca**

## **Atividades Práticas Supervisionadas Ciências da computação 4º semestre**

Carlos Eduardo dos Santos Ferreira – N6401C7

Gabriel Menezes de Antonio - F13GJI6

Gustavo Henrique dos Santos Faria – F22IFG2

Mayara Marques Pereira de Souza - N542DD1

## Índice

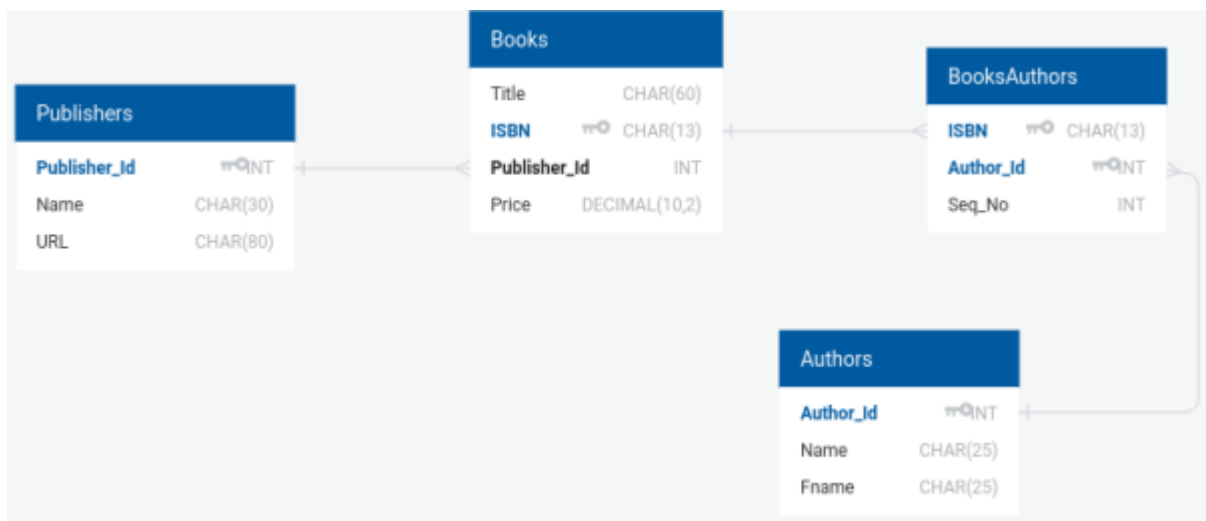
Objetivo do trabalho .....	3
Introdução .....	4
Referencial Teórico .....	6
Desenvolvimento.....	10
Resultados e Discussão .....	21
Considerações Finais .....	30
Referências Bibliográficas .....	31
Código fonte.....	33
Ficha de Atividades Práticas Supervisionadas .....	141

## **Objetivo do trabalho**

O objetivo desse documento é documentar as etapas realizadas durante o desenvolvimento do trabalho de APS no 4º semestre do curso de Ciências da computação, UNIP 2021.

## Introdução

Este trabalho teve como proposta inicial o desenvolvimento de um projeto com o tema “Desenvolvimento de sistema para manipulação de biblioteca”, onde o sistema a ser desenvolvido era o de uma livraria responsável por divulgar artigos científicos e livros relacionados à ciência e ao meio ambiente. O projeto deveria contar também com um banco de dados utilizado para a persistência de informações como livros, editoras e autores.



*MER do banco de dados proposto para o desenvolvimento*

Para o desenvolvimento desse projeto nos utilizamos de alguns padrões, bibliotecas e ferramentas, portanto para melhor entendimento das etapas a serem descritas no decorrer do trabalho uma sucinta definição de tais ferramentas foram introduzidas abaixo:

- **Java:** linguagem de programação orientada a objetos utilizada para o desenvolvimento do projeto.
- **NetBeans:** IDE (Ambiente de desenvolvimento integrado) para a linguagem Java utilizada no projeto.
- **Swing:** Java Swing é uma biblioteca de interface gráfica com o usuário (GUI) leve que inclui um rico conjunto de componentes. Faz parte das *Java Foundation Classes* (JFC) e inclui vários pacotes para o desenvolvimento de aplicativos de desktop Java (TECHOPEDIA, 2011).
- **MySQL:** MySQL é um sistema de gerenciamento de banco de dados relacional (RDBMS) de open-source da linguagem SQL (MYSQL REFERENCE MANUAL,

2021), essa sendo uma linguagem de programação padronizada utilizada para gerenciar, relacionar e executar diversas operações em banco de dados.

- **JDBC:** *Java Database Connectivity* é uma **API** (Interface de programação de aplicações) para a linguagem de programação Java, que define como um cliente pode acessar um banco de dados (THOMAS, 2002, p. 07). No caso deste projeto foi utilizada para fazer a conexão com um banco de dados MySQL.
- **DAO:** *Data Access Object*, ou em português, Objeto de acesso a dados é um padrão estrutural que nos permite isolar a camada de aplicação da camada de persistência (nesse caso um banco de dados relacional) usando uma API abstrata (THOMAS, 2002, p. 355).
- **MVC:** *Model-View-Controller* ou em português: Modelo-Visão-Controle é um padrão de projeto que separa conceitos lógicos de um projeto em três elementos interconectados com o intuito de facilitar a modificação e customização de cada parte (STELTING, 2002, p. 207). Dessa forma durante o desenvolvimento da aplicação as classes são separadas entre:
  - *Models* – Modelos de objetos Java;
  - *Views* – Classes de que fazem o *output* das informações contidas em objetos;
  - *Controllers* – Classes que fazem o controle das informações entre Models e Views, ou seja, Controllers recebem dados de objetos e os utiliza para atualizar as Views;

## Referencial Teórico

Durante o desenvolvimento do projeto várias etapas foram realizadas, indo da criação de *views* ao desenvolvimento do banco, porém o lugar onde “a magia acontece” é com certeza a etapa de processamento de dados. No projeto desenvolvido o processamento de dados é realizado através de algumas classes separadas em 2 pacotes, esses sendo: “*model.dao*” e “*controller*”.

No pacote “*model.dao*” estão presentes as classes de acesso a dados que, como o nome do pacote deixa evidente, estão padronizadas em formato DAO. Esse modo de padronização foi escolhido no desenvolvimento do projeto por diversos motivos, porém o principal deles sendo que o padrão DAO permite uma migração e manutenção mais fácil, uma vez que, o acesso aos dados é encapsulado e a alteração de algoritmos ou mesmo tipos de armazenamento (ex. mudança no tipo de banco de dados usado) é simplificada (THOMAS, 2002, p. 356).

Falando mais especificamente das classes DAO, elas são as responsáveis pelas operações de criação, leitura, atualização e deleção que serão efetuadas durante a aplicação, abaixo temos um método de leitura retirado da classe “*AuthorDAO.java*” como exemplo:

```
public static List<Author> getAuthors() {
    List<Author> authors = new ArrayList<Author>();
    try(Connection con = DriverManager.getConnection(URL, USER,
    PASS)) {
        String query = "SELECT * FROM authors";
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        while(rs.next()) {
            Integer id = rs.getInt("author_id");
            String name = rs.getString("name");
            String fname = rs.getString("fname");
            Author author = new Author(id, name, fname);
            authors.add(author);
        }
        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
    return authors;
}
```

No método “*getAuthors()*” inicialmente, uma Lista de autores é criada para armazenar os autores que serão recebidos do banco de dados, em seguida uma variável do tipo “*Connection*” é instanciada dentro um bloco *try*, com o intuito de tratar possíveis erros durante o acesso ao banco de dados sem que mensagens técnicas ou confusas sejam apresentadas ao usuário (PEREIRA, 2007).

Após ser estabelecida a conexão com o banco de dados, criamos uma *query* em SQL e o executamos usando um objeto da classe *ResultSet*. Esse objeto recebe os dados requeridos na *query*, estes sendo inseridos na Lista de autores inicialmente criada através de um *while*. Uma vez que todas as informações foram adicionadas a lista e o loop do *while* termina, a conexão com o banco de dados é encerrada e a lista de autores retornada.

Grande parte das classes do pacote *model.dao* são similares a esta apresentada, tendo como diferença apenas as operações que são realizadas (criação, leitura, atualização e deleção) e as informações que são manipuladas (livros, autores, editoras), exceto por uma classe, essa sendo “*ViewModelsDAO.java*”. Esta classe, diferente das outras citadas, contém métodos utilizados para uma *view* específica de busca, onde existe a opção de filtragem por título, autores ou editoras de um livro. Como exemplo temos o método “*getSearchByTitle()*”:

```
public static List<SearchViewModel> getSearchByTitle(String
bookTitle) {
    List<SearchViewModel> searchViewModels = new
        ArrayList<SearchViewModel>();

    try (Connection con = DriverManager.getConnection(URL, USER,
PASS)) {
        String query = "SELECT books.price, books.title, CONCAT
(authors.name,\" \",authors.fname) AS author,
publishers.name AS publisher, books.isbn\n" +
                        "FROM books INNER JOIN publishers INNER
JOIN booksauthors INNER JOIN authors\n" +
                        "ON publishers.publisher_id =
books.publisher_id\n" +
                        "AND booksauthors.isbn =books.isbn\n" +
                        "AND booksauthors.author_id =
authors.author_id\n" +
                        "WHERE books.title LIKE ? ";
        PreparedStatement stmt = con.prepareStatement(query);
        stmt.setString(1, "%" + bookTitle + "%");
        ResultSet rs = stmt.executeQuery();
        while(rs.next()) {
            Double price = rs.getDouble("price");
            String title = rs.getString("title");
```

```

        String author = rs.getString("author");
        String publisher = rs.getString("publisher");
        String isbn = rs.getString("isbn");

        SearchViewModel vm = new SearchViewModel(price, title,
        author, publisher, isbn);
        searchViewModels.add(vm);
    }
    con.close();
} catch (SQLException e) {
    e.printStackTrace();
}
return searchViewModels;
}

```

Este método possui várias semelhanças do método apresentado anteriormente sendo algumas: são utilizadas listas para o recebimento de dados, a classe *Connection* realiza a conexão com o banco de dados, a classe *ResultSet* executa uma query do tipo *PreparedStatement* no banco de dados.

Porém também se destacam algumas diferenças, sendo as principais delas a sua *query* SQL mais complexa, que conta com uma cláusula “INNER JOIN” utilizada para referenciar outras tabelas do banco de dados (JOIN CLAUSE, 2021). Ademais, por esse método ser utilizado para a pesquisa de livros baseado em seus respectivos títulos, temos uma cláusula “WHERE” na query, que filtra os resultados de acordo com o título inserido pelo usuário, é interessante também apontar que o método “*setString*” da classe *PreparedStatement* é utilizado para fazer a substituição da “?” na query pelo título inserido (PREPAREDSTATEMENT, 2021).

Já no pacote “controller” existe apenas uma classe “ControllerView.java”, essa tendo o intuito de relacionar as operações efetuadas nas classes *DAO* com as Views da aplicação. Este pacote foi criado a partir do uso do padrão MVC, que de acordo com Stelting (2002, p. 210) “[...] é um padrão que encoraja o bom encapsulamento”. Mais especificamente sobre o conceito de *Controller* temos que:

Este componente gerencia as mudanças ao *model*. Ele mantém uma referência ao componente do *model* que é responsável por realizar a mudança, enquanto o *controller* chama um ou mais métodos de atualização. Estas solicitações de mudança normalmente vêm de uma *view* (STELTING 2002, p. 212, traduzido do inglês).

Um exemplo de método de leitura é o “*readTablePublisher()*” apresentado abaixo:



```

public static void readTablePublisher() {
    DefaultTableModel modelo = (DefaultTableModel)
        ViewPublisher.tablePublisher.getModel();
    modelo.setNumRows(0);
    PublisherDAO pdao = new PublisherDAO();

    for (Publisher publisher: pdao.getPublishers()) {
        modelo.addRow(new Object[] {
            publisher.getId(),
            publisher.getName(),
            publisher.getUrl()
        });
    }
}

```

Este método tem uma lógica relativamente simples, inicialmente um objeto da classe *DefaultTableModel* é instanciado, este sendo referenciado a uma tabela dentro da *view* “ViewPublisher.java”, onde as editoras são apresentadas. A seguir, a classe “PublisherDAO.java” é instanciada com o intuito de ler todos as editoras presentes no banco de dados, para então esta lista ser adicionada à tabela mencionada, assim finalizando a execução do método.

## Desenvolvimento

O projeto de foi desenvolvido utilizando a linguagem de programação Java. Esta linguagem, por mais que seja antiga (O Java foi criado por volta de 1995, de acordo com Binstock, 2015), foi escolhida devido à diversas vantagens que foram um diferencial quanto outras opções como (SCHILDT, 2006, p. 9-11):

- Simplicidade: Java foi projetado para ser fácil para o programador profissional aprender e usar com eficácia. Supondo que você tenha alguma experiência em programação, não achará o Java difícil de dominar.
- Segurança: Java é considerada uma linguagem de programação mais segura porque não possui o conceito de ponteiros, o Java possui recursos interativos que podem ser embutidos em aplicativos da web. Esses recursos interativos em Java não permitem o acesso a outras partes do computador, o que o mantém longe de programas prejudiciais, como vírus e acesso não autorizado.
- Portabilidade: A portabilidade é um dos principais recursos do java, que permite que os programas java sejam executados em qualquer computador ou sistema operacional. Por exemplo, uma aplicação desenvolvida com Java é executada em uma ampla variedade de CPUs, sistemas operacionais e navegadores conectados à Internet.
- Orientada a objetos: Java é considerada uma linguagem de programação orientada a objetos pura. Em java, tudo é um objeto. Ele oferece suporte a todos os recursos do paradigma de programação orientada a objetos. Os tipos de dados primitivos também são implementados como objetos, mas ainda têm permissão para arquivar de alto desempenho.

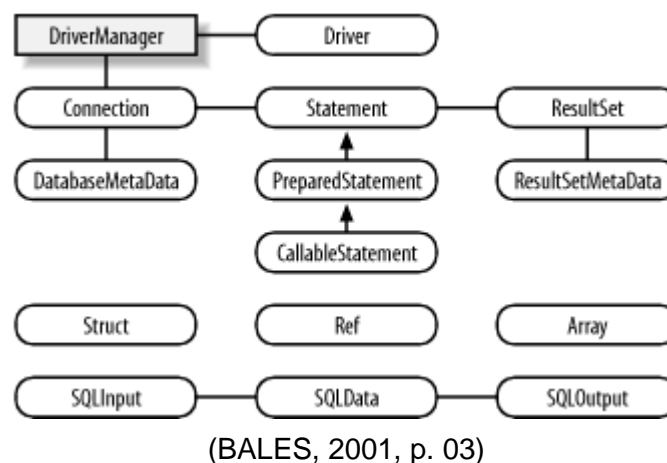
A IDE utilizada no desenvolvimento de projeto foi o NetBeans, por conta de sua integração com o Java. O NetBeans é um ambiente de desenvolvimento integrado (IDE) de código aberto gratuito que nos permite desenvolver aplicativos de desktop, móveis e web, além de fornecer suporte integrado para todo o ciclo de desenvolvimento, desde a criação do projeto até a depuração, criação de perfil e implantação.

Ao iniciarmos o projeto separamos o projeto em 3 etapas de desenvolvimento, algoritmos, views e conexão com banco de dados. Dessa forma, uma parte do grupo foi responsável pela construção e manutenção das views enquanto os demais ficaram

responsáveis pelo desenvolvimento da conexão com o banco de dados e os algoritmos relacionados ao banco.

Antes de iniciar uma **conexão entre o Java e um banco de dados** precisamos primeiro entender o conceito de JDBC. Nas palavras de Bales (2001, p. 03, traduzido do inglês) “JDBC é uma API Java usada para executar instruções SQL dinâmicas.”, ou seja, JDBC é basicamente uma API que conecta o Java a um banco de dados SQL.

A API JDBC é baseada principalmente em um conjunto de interfaces, não classes. Cabe ao fabricante do driver implementar as interfaces com seu próprio conjunto de classes (BALES, 2001, p. 03). O diagrama de classes abaixo mostra as classes e interfaces JDBC básicas;



(BALES, 2001, p. 03)

O banco de dados utilizado no projeto foi o MySQL, esse sendo um sistema de gerenciamento de banco de dados relacional de código aberto (RDBMS) baseado em Structured Query Language (SQL). O MySQL é baseado em um modelo cliente-servidor, o servidor MySQL lida com todas as instruções (ou comandos) do banco de dados. O servidor MySQL está disponível como um programa separado para uso em um ambiente de rede cliente-servidor e como uma biblioteca que pode ser incorporada em aplicativos separados (MOORE, 2018).

O processo para a conexão com o banco de dados foi feito através das classes *Connection* e *DriverManager* e o WampServer 3.2.3. Seguindo a documentação oficial encontrada no site do MySQL (JDBC Concepts, 2021) sabemos que para realizar a conexão entre o banco de dados e a aplicação java precisamos criar um objeto do tipo

Connection através do método “getConnection(URL, USER, PASS)”, da classe *DriverManager*, onde as variáveis URL, USER e PASS são, respectivamente, o endereço do banco, o usuário usado para acessar o banco de dados e a senha do usuário.

Dessa forma criamos a classe “DatabaseConstants.java”, com o intuito de manter essas informações essenciais para o acesso ao banco separada do resto da aplicação, assim caso alguma dessas informações precise ser atualizada, apenas uma porção do código precisará ser editada. Classe “DatabaseConstants.java”:

```
public class DatabaseConstants {
    public static final String URL =
        "jdbc:mysql://localhost/livraria_amazonia";
    public static final String USER = "root";
    public static final String PASS = "";
}
```

Uma vez que as constantes estão definidas podemos criar a conexão com o banco de dados, para exemplificar essa conexão vamos utilizar o método “getBook(String isbn)” da classe “BookDAO.java”:

```
1 public static Book getBook(String isbn) {
2     try (Connection con = DriverManager.getConnection(URL, USER,
3         PASS)) {
4         String query = "SELECT * FROM books WHERE isbn = ?";
5         PreparedStatement pstmt = con.prepareStatement(query);
6         pstmt.setString(1, isbn);
7         ResultSet rs = pstmt.executeQuery();
8
9         rs.next();
10        String title = rs.getString("title");
11        Integer publisherId = rs.getInt("publisher_id");
12        Double price = rs.getDouble("price");
13
14        return new Book(title, isbn, publisherId, price);
15    } catch (SQLException e) {
16        e.printStackTrace();
17        return null;
18    }
```

A conexão entre a aplicação e o banco de dados ocorre na 2ª linha do método, quando a classe *DriverManager* chama o método “getConnection” e cria um objeto da classe *Connection*, a partir deste momento temos uma conexão estabelecida com o banco. Na 3ª linha uma criamos uma instrução, ou **query**, SQL que seleciona todas

as colunas da tabela “books” onde o ISBN seja igual ao input do usuário, ou seja, esta query vai buscar os livros com o valor de ISBN inserido pelo usuário.

Após a criação da query, precisamos substituir a “?” na mesma pelo valor do ISBN inserido e executar a mesma. Estas operações são feitas nas linhas 5 e 6 respectivamente, onde na linha 4 a query é preparada, na 5 a substituição é efetuada e na 6, a query é finalmente executada. Uma vez que a query foi executada e a leitura feita podemos usar a variável “rs” para receber os valores lidos do banco de dados. Em suma este é o **processo de acesso aos dados** no sistema desenvolvido.

Tirando um pouco o foco do banco de dados e olhando mais para a aplicação, também foi necessário a criação de algumas classes “*bean*”, mais especificamente, uma para cada entidade presente no sistema. O intuito dessas classes é encapsular as informações a serem transportadas no sistema. Todas as classes *bean* foram organizadas num pacote chamado “model.bean”. Exemplo de classe *bean*:

```
public class Publisher {
    private int id;
    private String name;
    private String url;

    public Publisher(String name, String url) {
        this.name = name;
        this.url = url;
    }

    public Publisher(int publisherId, String name, String url) {
        this.id = publisherId;
        this.name = name;
        this.url = url;
    }

    public Publisher() {
    }

    public int getId() {
        return id;
    }

    public void setId(int publisherId) {
        this.id = publisherId;
    }

    public String getName() {
        return name;
    }
}
```

```

public void setName(String name) {
    this.name = name;
}

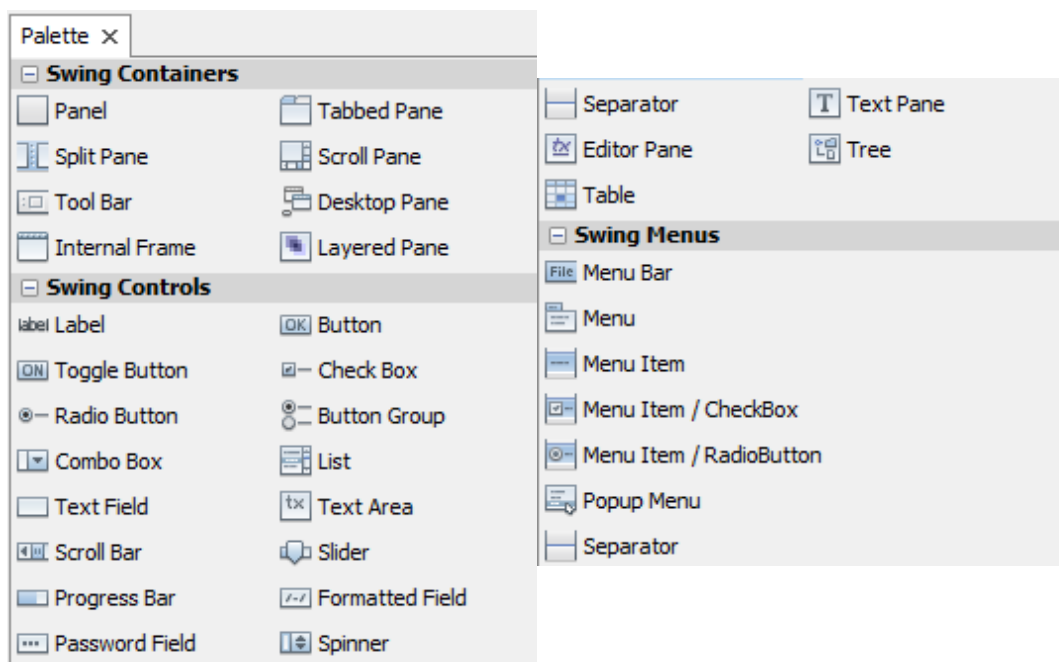
public String getUrl() {
    return url;
}

public void setUrl(String url) {
    this.url = url;
}

@Override
public String toString() {
    return getName();
}
}

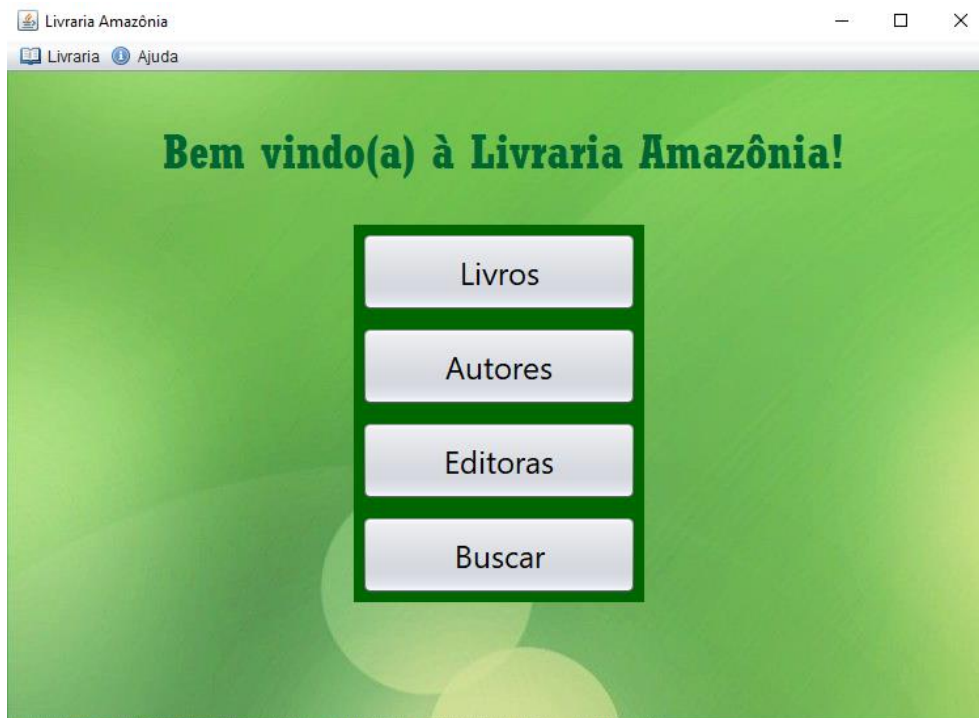
```

Para o desenvolvimento da interface a biblioteca swing do java foi utilizada. A biblioteca swing é composta por um conjunto de elementos de interação tais como janelas, botões, menus, ícones, barras de rolagem etc.



Além do uso destes elementos, também utilizamos imagens e ícones para deixar o projeto mais intuitivo, enquanto esteticamente mais bonito e minimalista. Essas imagens foram adicionadas ao projeto dentro de um pacote chamado "images", ao total 14 imagens foram adicionadas ao projeto.

Começando pela página inicial, optamos por um design mais limpo, apresentando o usuário com 4 botões: Livros, Autores, Editoras e Buscar. Os 3 primeiros botões estão relacionados com suas respectivas entidades do banco de dados, enquanto o botão de buscar é utilizado para facilitar a vida do usuário na busca por livros específicos. Ademais temos 2 menus de *dropdown* na parte superior da interface, o primeiro dando acesso aos mesmos componentes que os botões enquanto o segundo apresenta duas opções: “Sobre” e “Sair”.



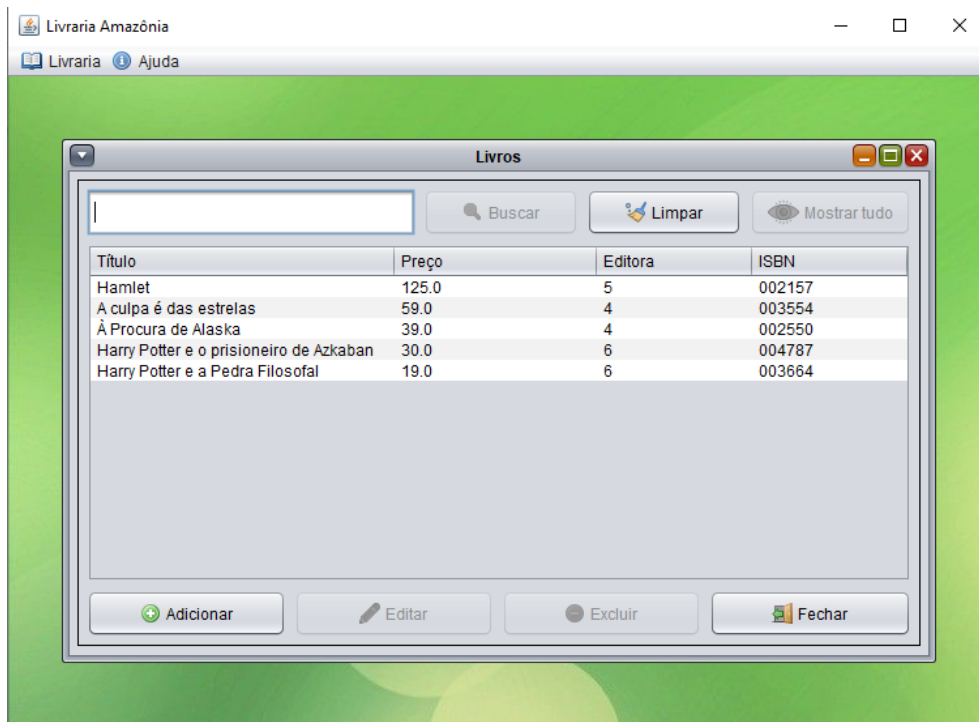
Janela inicial da aplicação



Menus *dropdown*

Ao clicar em livros uma nova guia irá se abrir para o usuário realizar operações com tal entidade. Interface de Livros foi dividida em três partes, na área central temos uma tabela exibindo todos os livros cadastrados, na área superior temos filtros

que podem ser usados para pesquisar livros, limpar a tabela, ou resetar a tabela (mostrar todos os livros novamente), e na parte inferior 4 botões, as três operações adicionar, editar e remover, e um quarto com a opção de fechar a janela.



Janela de visualização da entidade **livros**

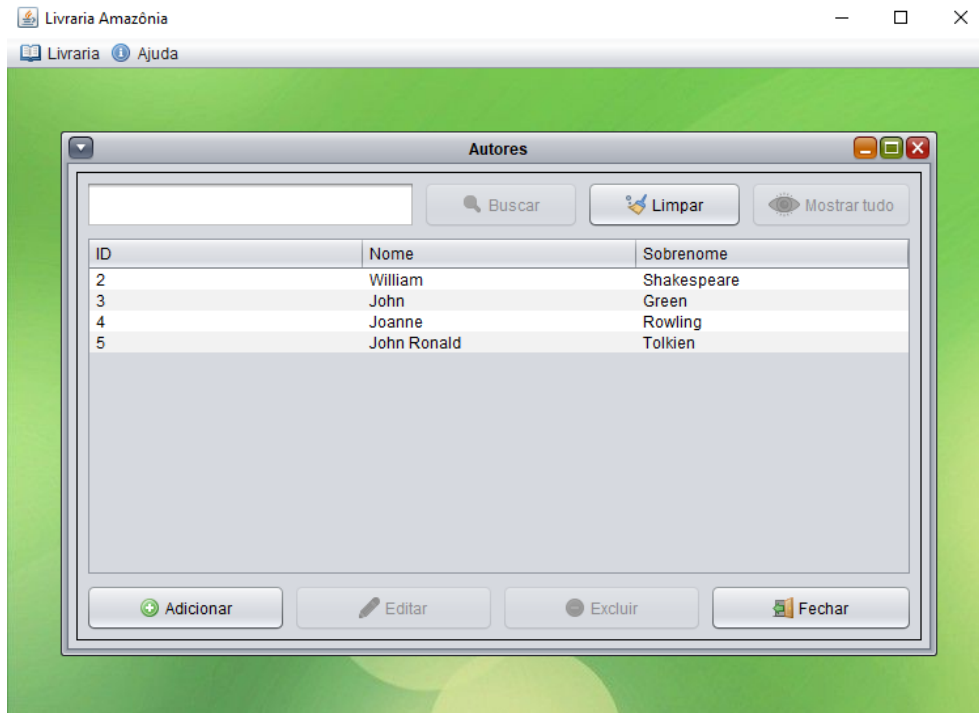
O **processo de visualização dos dados** nesse caso ocorre dentro do próprio método construtor da classe ao ser chamada. Dentro do método construtor o método “readTableBook()” da classe “ControllerView.java” é chamada, método exibido abaixo:

```
public static void readTableBook() {  
    DefaultTableModel modelo = (DefaultTableModel)  
        ViewBook.tableBook.getModel();  
    modelo.setNumRows(0);  
    BookDAO pdao = new BookDAO();  
  
    for (Book book: pdao.getBooks()) {  
        modelo.addRow(new Object[] {  
            book.getPrice(),  
            book.getTitle(),  
            book.getPublisherId(),  
            book.getIsbn()  
        });  
    }  
}
```

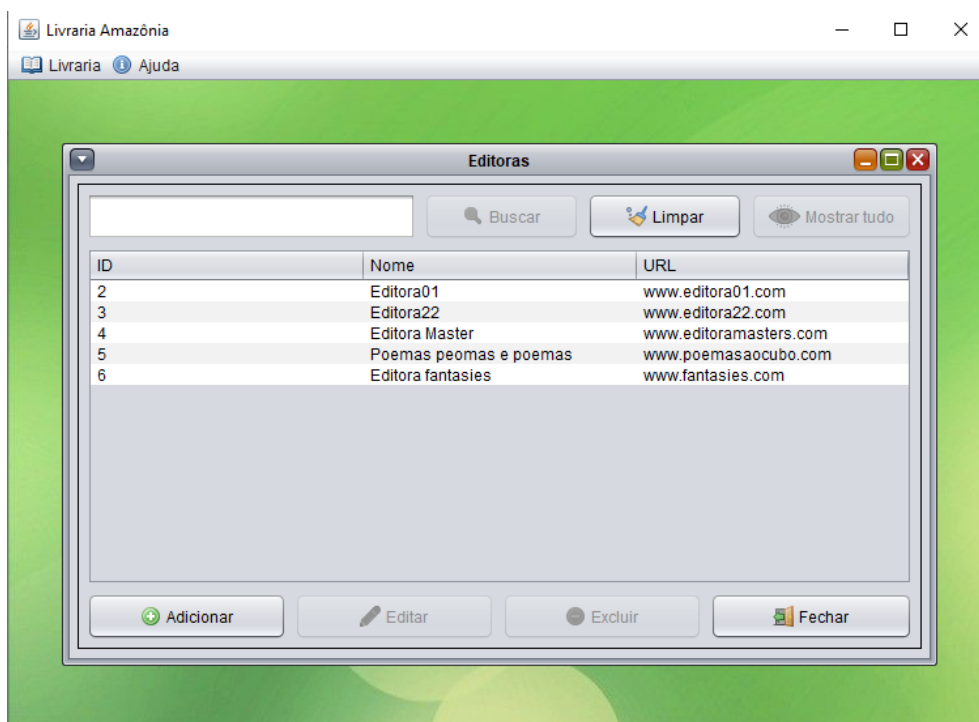


Uma vez que a os dados do banco são lidos através do método “getBooks()” da classe “BookDAO.java” as informações contidas na lista são inseridas na tabela uma linha por vez através de um ‘for’.

As interfaces de visualização de dados das entidades de Autores e Editoras tem designs e algoritmos muito semelhantes aos da entidade livro apresentado:

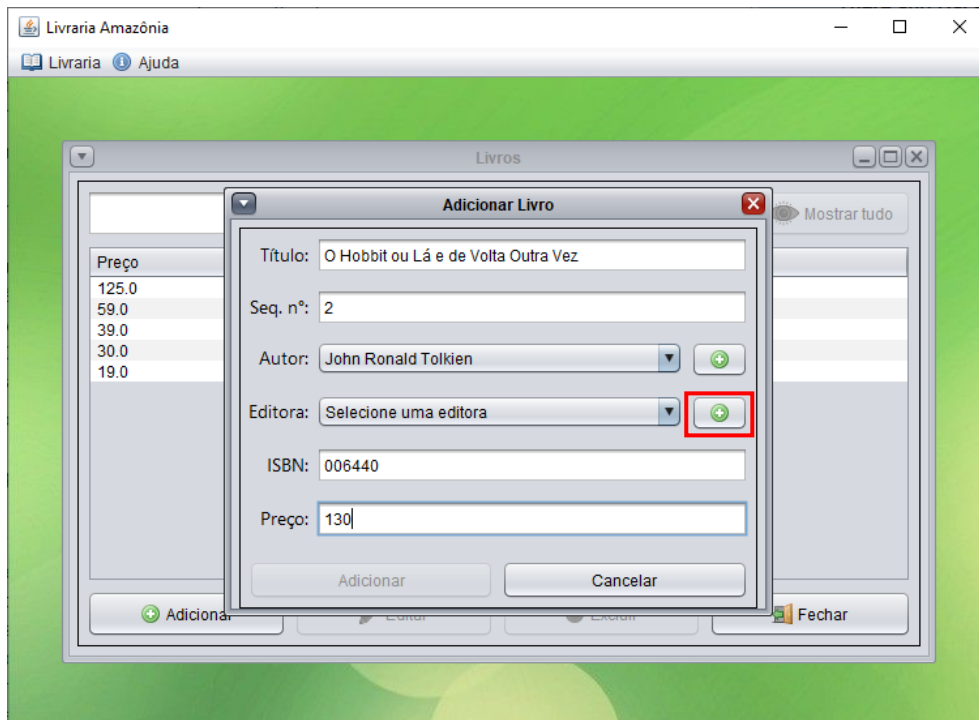


Janela de visualização da entidade **Autores**



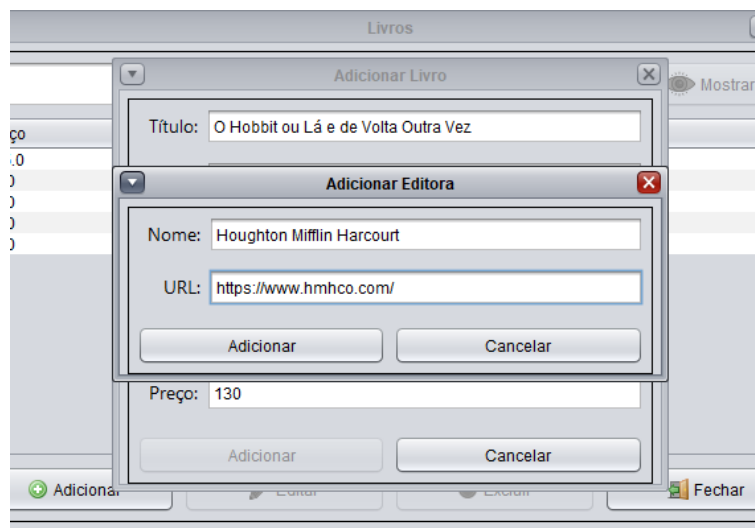
Janela de visualização da entidade **Editoras**

Para a operação de adicionar Livros uma nova interface foi criada, ela foi construída usando os campos obrigatórios para a criação de um livro, porém um botão ao lado dos dropdowns de seleção de Autores e Editoras foi adicionado. Esse botão é usado para adicionar um novo(a) Autor/Editora se ter que sair da janela. O intuito dessa adição é deixar o uso da aplicação mais dinâmica e intuitiva, uma vez que caso o usuário entre na interface de adição de livro, porém ainda não tenha adicionado os respectivos Autores/Editoras desse livro ele pode fazê-lo sem mudar de janela.

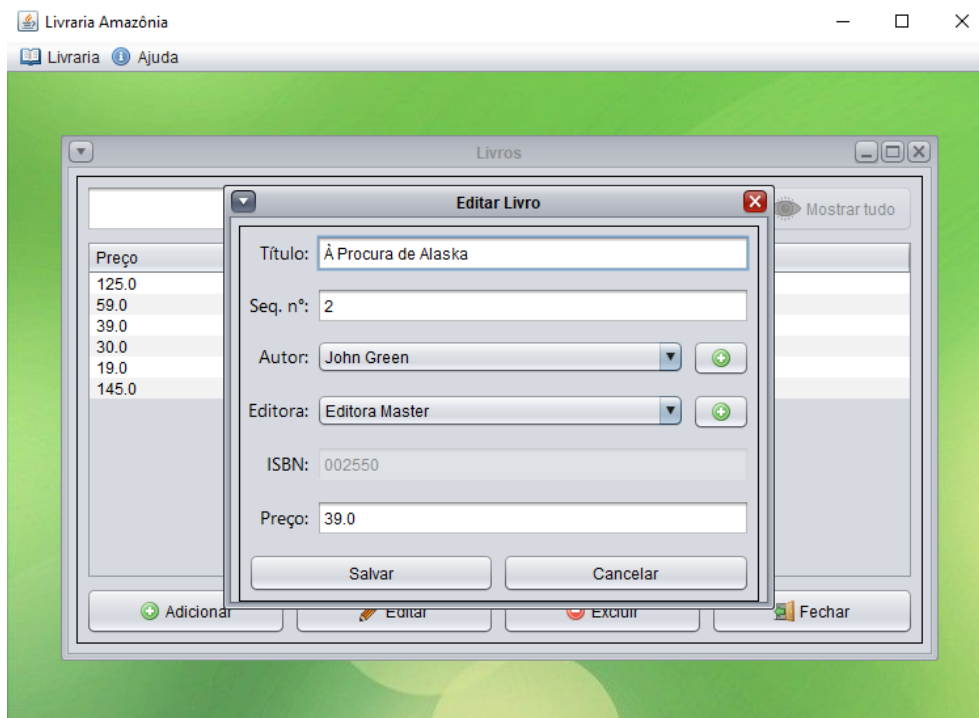


Janela de criação de novo **livro**  
(botão mencionado destacado em vermelho)

Clicando no botão mencionado a janela de criação de nova Editora é chamada:



A interface de edição de livros é bem semelhante com a de adição, tendo como diferença apenas o fato de que o ISBN não é editável, visto que ele é a chave primário do livro durante o cadastro:



As interfaces de Autores e editores são uma simples janela solicitando as informações para a criação/edição da entidade:

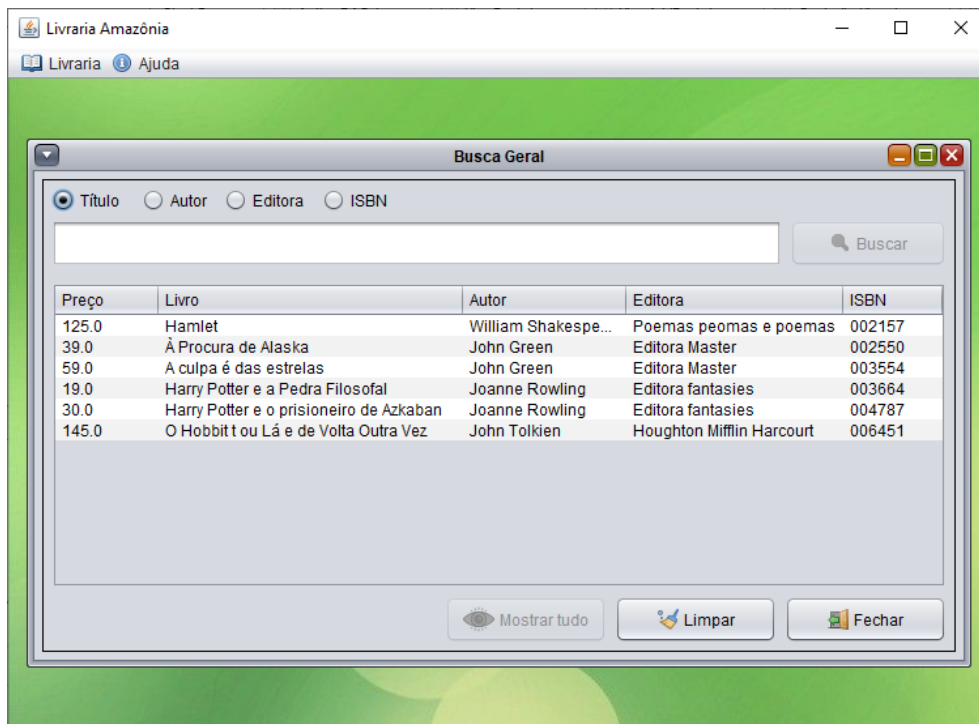
A screenshot of a dialog box titled 'Adicionar Autor'. It has two input fields: 'Nome:' and 'Sobrenome:'. Below the fields are two buttons: 'Adicionar' and 'Cancelar'.

Janela de criação de Autor

A screenshot of a dialog box titled 'Adicionar Editora'. It has two input fields: 'Nome:' and 'URL:'. Below the fields are two buttons: 'Adicionar' and 'Cancelar'.

Janela de criação de Editora

Por fim, mas não menos importante, temos a interface Busca. Criamos essa interface com o objetivo de facilitar a visualização do usuário na relação entre as entidades, ou seja, fica evidente a relação entre um livro e autor ou editora, uma vez que todos esses elementos aparecem em uma mesma tabela.



Janela de visualização da área de busca

Além da tabela, temos uma área para filtragem na parte superior da janela, onde o usuário pode buscar por livros filtrando-os por título, autor, editora ou ISBN. O processo de visualização de dados para essa interface é exatamente igual ao realizado nas demais. A única diferença esta tabela e as demais é a query SQL utilizada, pois a query utilizada nesta interface é mais complexa que as demais:

```
SELECT books.price, books.title, CONCAT
(authors.name, "\", \"", authors.fname) AS author,
publishers.name AS publisher, books.isbn FROM books
INNER JOIN publishers INNER JOIN booksauthors INNER JOIN authors
ON publishers.publisher_id = books.publisher_id
AND booksauthors.isbn = books.isbn
AND booksauthors.author_id = authors.author_id
```

## Resultados e Discussão

Os testes do trabalho para analisar o resultado final foram feitos em 3 etapas:

1. **Teste principal:** nessa etapa, testamos se todos os métodos acerca do objetivo do trabalho relacionados a cadastro e visualização dos dados, estavam funcionando corretamente, caso fossem inseridos os dados corretos em cada um deles.
2. **Teste de controle de erros na entrada:** já nessa etapa, com o teste principal e seus ajustes feitos, o programa já está rodando corretamente, mas temos alguns detalhes a tratar: e se o usuário inserir algo que não condiz com o que estamos pedindo? O que será retornado para ele? Para onde será redirecionado? O objetivo é garantir que nesses casos, o programa não finalize de repente, não entre em algum loop, ou dê qualquer tipo de problema sem solução ou explicação ao usuário.
3. **Teste final:** é uma última verificação de que nada escapou dos dois testes anteriores, basicamente, testamos tudo novamente, testando o máximo de possibilidades de entradas corretas e incorretas, para entregarmos tudo devidamente funcionando, sem possibilidade de ocorrer erros inesperados.

Para a etapa de **testes principais** vamos seguir os requerimentos pedidos para o desenvolvimento da aplicação, estes sendo:

- Listar (e mostrar) através de um sistema de busca: Livros, Autores e Editoras
- Incluir: Livros, Autores e Editoras
- Modificar: Livros, Autores e Editoras
- Excluir: Livros, Autores e Editoras

- **Listagem de autores ✓** – Clicando no botão “Autores” do menu inicial uma tabela listando todos os autores cadastrados é exibida, caso o usuário queira filtrar os autores ele pode digitar numa caixa de texto acima da tabela e clicar no botão “Buscar”. O filtro busca tanto pelo nome quanto pelo ID o autor.

The screenshot shows a window titled "Autores". At the top, there is a search bar containing the text "John". To the right of the search bar are three buttons: "Buscar" (with a magnifying glass icon), "Limpar" (with a trash can icon), and "Mostrar tudo" (with an eye icon). Below the search bar is a table with three columns: "ID", "Nome", and "Sobrenome". The table contains two rows of data. At the bottom of the window, there are four buttons: "Adicionar" (with a plus icon), "Editar" (with a pencil icon), "Excluir" (with a minus icon), and "Fechar" (with a close icon).

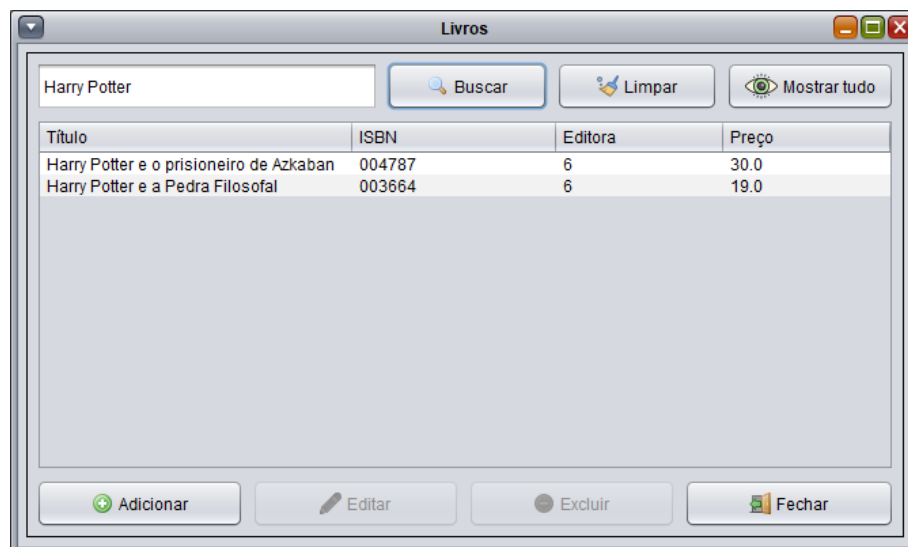
ID	Nome	Sobrenome
3	John	Green
5	John	Tolkien

- **Listagem de editoras ✓** – Clicando no botão “Editoras” do menu inicial uma tabela listando todas as editoras cadastradas é exibida, caso o usuário queira filtrar as editoras ele pode digitar numa caixa de texto acima da tabela e clicar no botão “Buscar”. O filtro busca pelo nome, URL e ID das editoras.

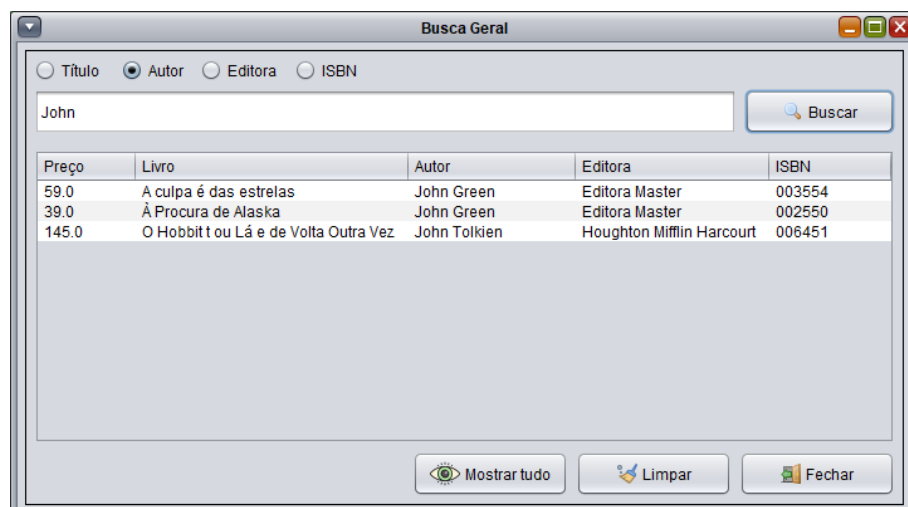
The screenshot shows a window titled "Editoras". At the top, there is a search bar containing the text "Editora". To the right of the search bar are three buttons: "Buscar" (with a magnifying glass icon), "Limpar" (with a trash can icon), and "Mostrar tudo" (with an eye icon). Below the search bar is a table with three columns: "ID", "Nome", and "URL". The table contains five rows of data. At the bottom of the window, there are four buttons: "Adicionar" (with a plus icon), "Editar" (with a pencil icon), "Excluir" (with a minus icon), and "Fechar" (with a close icon).

ID	Nome	URL
2	Editora 01	www.editora01.com
3	Editora 22	www.editora22.com
4	Editora Master	www.editoramasters.com
6	Editora fantasies	www.fantasies.com

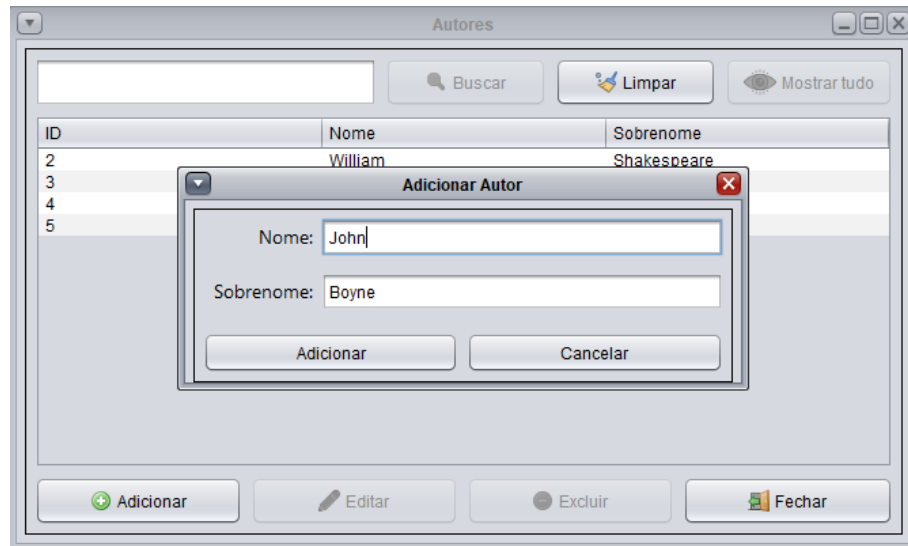
- **Listagem de livros ✓** – Clicando no botão “Livros” do menu inicial uma tabela listando todos os livros cadastrados é exibida, caso o usuário queira filtrar os livros ele pode digitar numa caixa de texto acima da tabela e clicar no botão “Buscar”. O filtro busca pelo título, ISBN, ID da editora e preço.



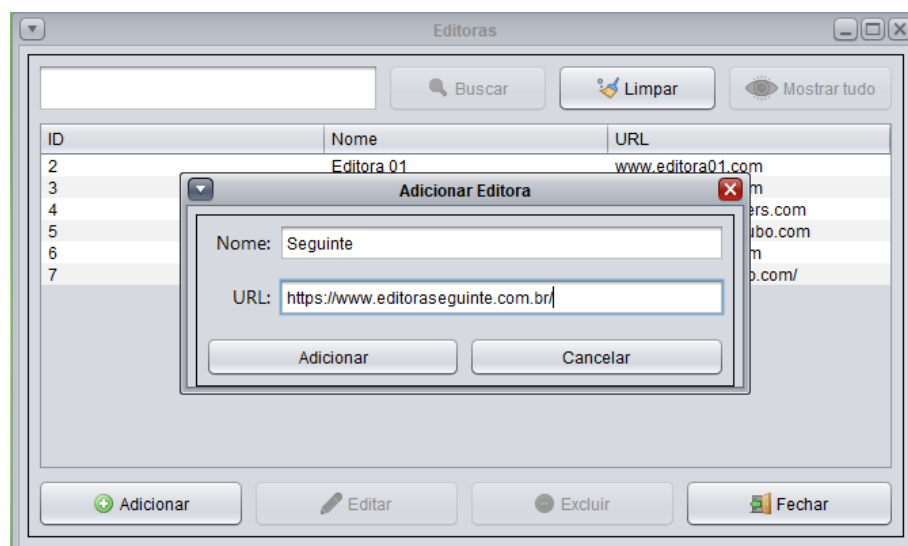
- **Listagem usando busca composta ✓** – Clicando no botão “Buscar” do menu inicial uma tabela listando todos os livros cadastrados é exibida, porém essa tabela não é composta apenas por livros, mas também por suas relações com autores e editoras. Caso o usuário queira filtrar os livros ele pode digitar numa caixa de texto acima da tabela, selecionar porque tipo de informação os livros serão filtrados e clicar no botão buscar.



- **Inserção de autores** ✓ – A inserção de autores é feita de forma bem simples e intuitiva, após entrar na área dos autores, clicamos no botão adicionar na parte inferior, então apenas é preciso preencher os campos de nome e sobrenome do autor e clicar em adicionar.



- **Inserção de editoras** ✓ – A inserção de editoras é feita de similar à de autores, após entrar na área de editoras, clicamos no botão adicionar na parte inferior, então apenas é preciso preencher os campos de nome e URL da editora e clicar em adicionar.





- **Inserção de livros ✓** – A inserção de livros segue o mesmo padrão da inserção de autores e editoras, clicamos no botão adicionar na parte inferior da área de livros, e então preenchemos os campos de título, sequência, ISBN e preço. Além disso, também é preciso selecionar o autor e editora do livro através de dois *dropdowns*, caso o autor ou editora ainda não tenha sido adicionado ao banco de dados, essa ação pode ser feita na própria janela, clicando no botão com símbolo de adição (“+”) ao lado do respectivo *dropdown*. Após todos os campos serem preenchidos basta clicar em no botão adicionar para inserir o livro.

- **Modificação de autores ✓** – A modificação/edição de autores é feita de forma similar a inserção, porém ao invés de clicar no botão “Adicionar”, deve-se selecionar uma linha da tabela exibida e clicar no botão “Editar”. Os campos não editáveis são as chaves primárias da linha selecionada.

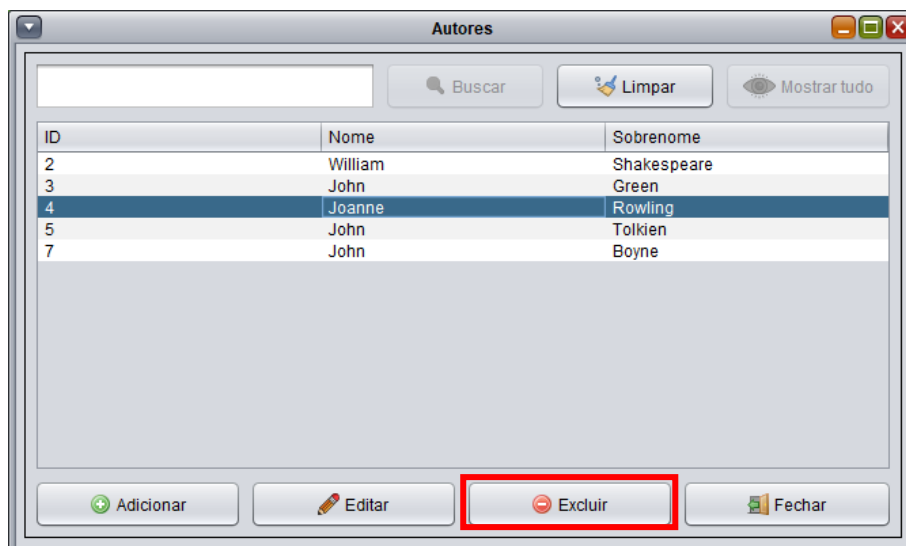
- **Modificação de editoras ✓** – A modificação/edição de editoras é feita de forma similar a inserção, porém ao invés de clicar no botão “Adicionar”, deve-se selecionar uma linha da tabela exibida e clicar no botão “Editar”. Os campos não editáveis são as chaves primárias da linha selecionada.

The image shows a dialog box titled "Editar Editora". It has three input fields: "ID:" with the value "8", "Nome:" with the value "Seguinte", and "URL:" with the value "https://www.editoraseguinte.com.br/". At the bottom of the dialog are two buttons: "Salvar" and "Cancelar".

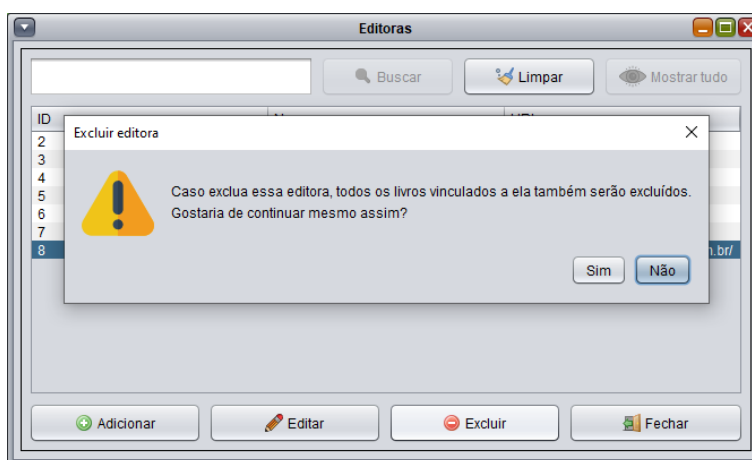
- **Modificação de livros ✓** – A modificação/edição de livros, assim como a de autores e editoras, é feita de forma similar a inserção, porém ao invés de clicar no botão “Adicionar”, deve-se selecionar uma linha da tabela exibida e clicar no botão “Editar”. Os campos não editáveis são as chaves primárias da linha selecionada.

The image shows a dialog box titled "Editar Livro". It has six input fields: "Título:" with the value "O menino do pijama listrado", "Seq. nº:" with the value "1", "Autor:" with the value "John Boyne", "Editora:" with the value "Seguinte", "ISBN:" with the value "001045", and "Preço:" with the value "44.9". At the bottom of the dialog are two buttons: "Salvar" and "Cancelar".

- **Exclusão de autores ✓** – A exclusão de autores é feita de forma extremamente simples, o usuário deve apenas selecionar um autor exibido na lista e clicar no botão excluir na parte inferior:

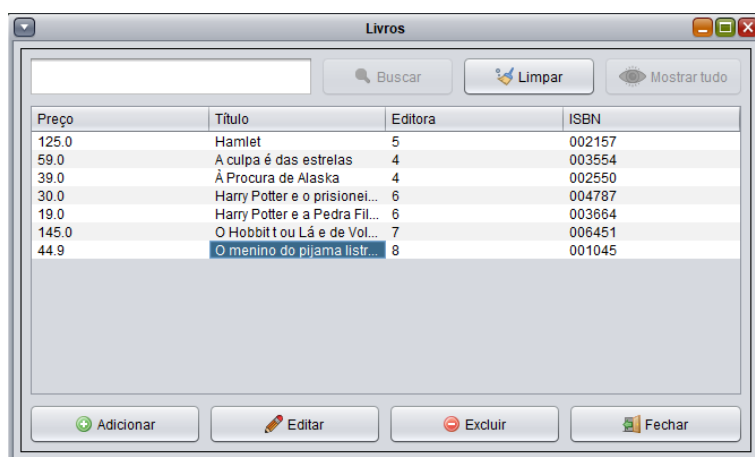


- **Exclusão de editoras** ✓ – A exclusão de editoras é feita de forma similar a de autores, porém após clicar-se no botão excluir, caso a editora tenha algum livro relacionada a si no banco de dados, um *popup* aparece avisando o usuário que caso a editora seja excluída seus livros também serão excluídos, dando a opção ao usuário se ele deseja prosseguir com a operação ou não.



- **Exclusão de livros** ✓ –

A exclusão de livros é feita de forma similar a de autores, o usuário apenas precisa selecionar o livro e clicar em excluir para realizar a operação.



Após verificarmos que todas as operações estavam funcionando corretamente seguimos para etapa de teste de controle de erros na entrada, durante essa etapa tentamos todos os tipos de input possíveis que poderiam causar erros na aplicação. Uma lista de algumas de nossas tentativas em causar erros na aplicação pode ser encontrada abaixo:

- **Adicionar autores, editoras ou livros faltando informações:** O botão de adicionar é apenas ativado depois que todos os campos obrigatórios são preenchidos;

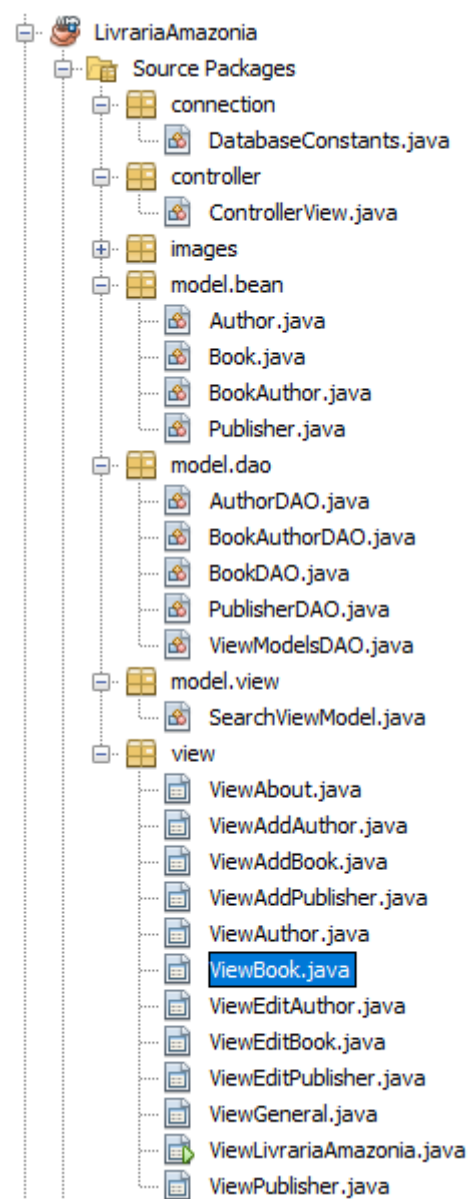
- **Preencher campos numéricos com letras ou símbolos:** Todos os campos que estão relacionados a valores que não são *strings* tem uma verificação embutida, fazendo com que apenas caracteres validos possam ser digitados, por exemplo, no campo de preço dos livros apenas números e 1 ponto (".") poder ser digitados.
- **Deletar uma editora relacionada a um livro:** Como mencionado na página anterior (27), quando o usuário tenta excluir uma editora um *popup* aparece avisando o usuário que caso a editora seja excluída seus livros também serão excluídos. Tivemos que implementar essa funcionalidade pois os livros tem editoras como chaves estrangeiras no banco de dados, então sua exclusão causaria erros relacionais no banco de dados.

Terminando a fase de testes concluímos que todas as funcionalidades funcionam corretamente graças aos métodos que criamos já contendo o controle de erros na entrada. Os exemplos mostrados nesse capítulo são apenas alguns de inúmeros testes que foram realizados até chegarmos nesse resultado, tivemos diversos problemas com o programa apresentando *bugs* ou parando inesperadamente, mas conseguimos contorná-los e deixamos tudo o mais fluído e intuitivo possível para o usuário final.

## Estrutura da aplicação

A aplicação foi dividida num total de sete pacotes, sendo estes: connection, controller, images, model.bean, model.dao, model.view, e view. Separando a aplicação em diversos pacotes deixamos o código muito organizado e limpo para uma fácil leitura de qualquer um que queira entendê-lo. Além disso essa estrutura de pacotes cria um bom encapsulamento de certas informações deixando a aplicação mais segura.

Acreditamos que a única desvantagem dessa estrutura seja que a quantidade de classes aumente de forma drástica, porém isso não é um problema em nossa opinião, uma vez que a quantidade de classes do projeto não pode afetar o mesmo negativamente de nenhuma forma.



## Considerações Finais

O grupo como um todo ficou muito satisfeito com o resultado do projeto no geral, conseguimos separar as cargas de uma forma em que ninguém ficasse sobrecarregado e finalizamos o projeto no prazo previsto. O desenvolvimento ocorreu sem imprevistos e chegamos ao final do projeto com um código organizado, legível, de fácil entendimento e seguindo as boas práticas ensinadas durante as aulas.

Uma das técnicas utilizadas que seguimos de forma rígida foi a implementação do padrão MVC. Durante o desenvolvimento separamos as classes em modelos, controladores e views, pois vimos nesse padrão as vantagens dessa separação e isso nos trouxe benefícios durante o desenvolvimento, visto que essa organização facilitou o entendimento do grupo como um todo do código.

Ademais no desenvolvimento das *view* escolhemos usar `JInternalFrame` dentro de `JDesktopPane`, para que as janelas fossem abertas de maneira mais natural, similar as janelas do Windows.

Em relação as interfaces, fomos os mais objetivo possíveis para que a aplicação ficasse intuitiva para o usuário, uma vez que quanto menos coisas na tela do usuário mais fácil fica o entendimento do mesmo. Além disso mantemos a interface resumida as entidades (autores, editoras e livros), deixando-a assim clara e objetiva.

A parte mais desafiadora do desenvolvimento foi fazer a integração entre o banco de dados e a aplicação java, uma vez que precisamos fazer muita pesquisa para entendermos como funcionava a API JDBC, porém uma vez que a integração foi feita o desenvolvimento fluiu sem problemas, já que não tivemos problemas em desenvolver queries SQL ou utilizar os retornos do banco de dados na aplicação.

## Referências Bibliográficas

SWING. In: TECHOPEDIA. Edmonton, AB: Janalta Interactive Inc, 2011. Disponível em: <https://www.techopedia.com/definition/26102/java-swing>. Acesso em: 09 nov. 2021.

THOMAS, Todd M. **Java Data Access**. New York, NY: M&T Books, 2002.

STELTING, Stephen; MAASSEN, Olav. **Applied Java Patterns**. Upper Saddle River, NJ: Prentice Hall, 2002.

PEREIRA, Rafael Cardoso. Blocos Try/Catch. **DevMedia**, 2007. Disponível em: <https://www.devmedia.com.br/blocos-try-catch/7339/>. Acesso em: 13 nov. 2021.

JOIN CLAUSE. **MySQL 8.0 Reference Manual**. Oracle Corporation, 2021. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/join.html/>. Acesso em: 15 nov. 2021.

WHAT IS MYSQL?. **MySQL 8.0 Reference Manual**. Oracle Corporation, 2021. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html/>. Acesso em: 15 nov. 2021.

PREPAREDSTATEMENT. **Java™ Platform, Standard Edition 8 API Specification**. Oracle Corporation, 2021. Disponível em: <https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html/>. Acesso em: 14 nov. 2021.

BINSTOCK, Andrew. **Java's 20 Years of Innovation**. Forbes, maio 2015. Arquivado em: <https://web.archive.org/web/20160314102242/http://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/>

SCHILDT, Herbert. **Java The Complete Reference, Seventh Edition**. McGraw-Hill Osborne, 2006.

JDBC Concepts. **MySQL Connector/J 5.1 Developer Guide**. Oracle Corporation, 2021. Disponível em: <https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-usagenotes-connect-drivermanager.html/>. Acesso em: 17 nov. 2021.

BALES, Donald. **Java Programming with Oracle JDBC**. O'Reilly Media, 2001.

MOORE, Lindsay. Blocos What is MySQL?. **SearchOracle**, 2018. Disponível em: <https://searchoracle.techtarget.com/definition/MySQL/>. Acesso em: 17 nov. 2021

Organização: Oracle. Título: **Class JTextField**. Ano: c1995 Disponível em: <https://docs.oracle.com/javase/7/docs/api/javax/swing/JTextField.html> Acesso em: 06 de out. de 2021

Organização: Oracle. Título: **Class JOptionPane**. Ano: c1995 Disponível em: <https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html> Acesso em: 06 de out. de 2021

Organização: Oracle. Título: **How to Use the ButtonGroup Component**. Ano: c1995 Disponível em: <<https://docs.oracle.com/javase/tutorial/uiswing/components/buttongroup.html>> Acesso em: 06 de out. de 2021

Organização: Oracle. Título: **How to Use Buttons, Check Boxes, and Radio Buttons**. Ano: c1995 Disponível em: <<https://docs.oracle.com/javase/tutorial/uiswing/components/button.html#radiobutton>> Acesso em: 06 de out. de 2021

Organização: Oracle. Título: **How to Use Icons**. Ano: c1995 Disponível em: <<https://docs.oracle.com/javase/tutorial/uiswing/components/icon.html>> Acesso em: 26 de out. de 2021

Organização: DevMedia. Título: **Conhecendo Gerenciadores de Layout GUI do Java**. Ano: 2012 Disponível em: <<https://www.devmedia.com.br/conhecendo-gerenciadores-de-layout-gui-do-java/25869>> Acesso em: 26 de out. de 2021

Organização: DevMedia. Título: **Utilizando JComboBox- Java Swing componentes NetBeans – Parte 11**. Ano: 2011 Disponível em: <<https://www.devmedia.com.br/utilizando-jcombobox-java-swing-componentes-netbeans-parte-11/22096>> Acesso em: 31 de out. de 2021

Organização: Oracle. Título: **How to Use Internal Frames**. Ano: c1995 Disponível em: <<https://docs.oracle.com/javase/tutorial/uiswing/components/internalframe.html>> Acesso em: 31 de out. de 2021

Organização: Oracle. Título: **Class JDesktopPane**. Ano: c1995 Disponível em: <<https://docs.oracle.com/javase/7/docs/api/javax/swing/JDesktopPane.html>> Acesso em: 31 de out. de 2021



## Código fonte

Para executar o código, por favor acessar o repositório do GitHub abaixo:

<https://github.com/Gustavo-tech/APS-4Semester>

### > Pacote connection:

DatabaseConstants.java

```
package connection;
```

```
public class DatabaseConstants {  
    public static final String URL = "jdbc:mysql://localhost/livraria_amazonia";  
    public static final String USER = "root";  
    public static final String PASS = "";  
}
```

### > Pacote controller:

ControllerView.java

```
package controller;
```

```
import javax.swing.JOptionPane;  
import javax.swing.table.DefaultTableModel;  
import model.bean.*;  
import model.dao.*;  
import model.view.*;  
import view.*;
```

```
public class ControllerView {  
  
    // exibe (id, nome, url) na tabela de editora na tabela  
    public static void readTablePublisher() {  
        DefaultTableModel modelo = (DefaultTableModel) ViewPublisher.tablePublisher.getModel();  
        modelo.setNumRows(0);  
        PublisherDAO pdao = new PublisherDAO();  
  
        for (Publisher publisher: pdao.getPublishers()) {  
            modelo.addRow(new Object[] {  
                publisher.getId(),  
                publisher.getName(),  
                publisher.getUrl()  
            });  
        }  
    }  
  
    // exibe (preço, título, autor, editora e isbn) de todos os livros cadastrados na tabela  
    public static void readTableGeneral() {  
        DefaultTableModel modelo = (DefaultTableModel) ViewGeneral.tableGeneral.getModel();  
        modelo.setNumRows(0);  
        ViewModelsDAO vmdao = new ViewModelsDAO();  
    }  
}
```

```

        for (SearchViewModel search: vmdao.getSearchViewModels()) {
            modelo.addRow(new Object[] {
                search.getTitle(),
                search.getIsbn(),
                search.getAuthor(),
                "R$" + String.format("%.2f", search.getPrice()),
                search.getPublisher()
            });
        }
    }

    // exibe (preço, título, autor, editora e isbn) dos livros da busca por título na tabela
    public static void readTableTitle(String title) {
        DefaultTableModel modelo = (DefaultTableModel) ViewGeneral.tableGeneral.getModel();
        modelo.setNumRows(0);
        ViewModelsDAO vmdao = new ViewModelsDAO();

        for (SearchViewModel search: vmdao.getSearchByTitle(title)) {
            modelo.addRow(new Object[] {
                search.getTitle(),
                search.getIsbn(),
                search.getAuthor(),
                "R$" + String.format("%.2f", search.getPrice()),
                search.getPublisher()
            });
        }

        if (modelo.getRowCount() == 0) {
            try {
                JOptionPane.showInternalMessageDialog(null, "Livro não encontrado.", "Aviso",
                JOptionPane.WARNING_MESSAGE );
                ViewGeneral.buttonClean.setEnabled(false);
            } catch (Exception ignored) {}
        }
    }

    // exibe (preço, título, autor, editora e isbn) dos livros da busca por autor na tabela
    public static void readTableAuthor(String author) {
        DefaultTableModel modelo = (DefaultTableModel) ViewGeneral.tableGeneral.getModel();
        modelo.setNumRows(0);
        ViewModelsDAO vmdao = new ViewModelsDAO();

        for (SearchViewModel search: vmdao.getSearchByAuthor(author)) {
            modelo.addRow(new Object[] {
                search.getTitle(),
                search.getIsbn(),
                search.getAuthor(),
                "R$" + String.format("%.2f", search.getPrice()),
                search.getPublisher()
            });
        }

        if (modelo.getRowCount() == 0) {
            try {

```

```

        JOptionPane.showInternalMessageDialog(null, "Autor(a) não encontrado.", "Aviso",
JOptionPane.WARNING_MESSAGE );
        ViewGeneral.buttonClean.setEnabled(false);
    } catch (Exception ignored) {}
}
}

// exibe (preço, título, autor, editora e isbn) dos livros da busca por editora na tabela
public static void readTablePublisher(String publisher) {
    DefaultTableModel modelo = (DefaultTableModel) ViewGeneral.tableGeneral.getModel();
    modelo.setNumRows(0);
    ViewModelsDAO vmdao = new ViewModelsDAO();

    for (SearchViewModel search: vmdao.getSearchByPublisher(publisher)) {
        modelo.addRow(new Object[] {
            search.getTitle(),
            search.getIsbn(),
            search.getAuthor(),
            "R$" + String.format("%.2f", search.getPrice()),
            search.getPublisher()
        });
    }

    if (modelo.getRowCount() == 0) {
        try {
            JOptionPane.showInternalMessageDialog(null, "Editora não encontrada.", "Aviso",
JOptionPane.WARNING_MESSAGE );
            ViewGeneral.buttonClean.setEnabled(false);
        } catch (Exception ignored) {}
    }
}

// exibe (preço, título, autor, editora e isbn) dos livros da busca por isbn na tabela na tabela
public static void readTableIsbn(String isbn) {
    DefaultTableModel modelo = (DefaultTableModel) ViewGeneral.tableGeneral.getModel();
    modelo.setNumRows(0);
    ViewModelsDAO vmdao = new ViewModelsDAO();

    for (SearchViewModel search: vmdao.getSearchByIsbn(isbn)) {
        modelo.addRow(new Object[] {
            search.getTitle(),
            search.getIsbn(),
            search.getAuthor(),
            "R$" + String.format("%.2f", search.getPrice()),
            search.getPublisher()
        });
    }

    if (modelo.getRowCount() == 0) {
        try {
            JOptionPane.showInternalMessageDialog(null, "ISBN não encontrado.", "Aviso",
JOptionPane.WARNING_MESSAGE );
            ViewGeneral.buttonClean.setEnabled(false);
        } catch (Exception ignored) {}
    }
}
}

```

```

// para trazer os dados de editora para a tabela
public static void readTableAuthor() {
    DefaultTableModel modelo = (DefaultTableModel) ViewAuthor.tableAuthor.getModel();
    modelo.setNumRows(0);
    AuthorDAO pdao = new AuthorDAO();

    for (Author author: pdao.getAuthors()) {
        modelo.addRow(new Object[] {
            author.getId(),
            author.getName(),
            author.getFName()
        });
    }
}

// para trazer os dados de livros para a tabela
public static void readTableBook() {
    DefaultTableModel modelo = (DefaultTableModel) ViewBook.tableBook.getModel();
    modelo.setNumRows(0);
    BookDAO pdao = new BookDAO();

    for (Book book: pdao.getBooks()) {
        modelo.addRow(new Object[] {
            book.getTitle(),
            book.getIsbn(),
            "R$" + String.format("%.2f", book.getPrice()),
            book.getPublisherId()
        });
    }
}

// coloca os autores no "comboBoxAuthor" na tela de edição de livro
public static void updateEditComboAuthor() {
    AuthorDAO dao = new AuthorDAO();
    ViewEditBook.comboBoxAuthor.removeAllItems();
    ViewEditBook.comboBoxAuthor.addItem("Selecione um autor(a)");
    for (Author author: dao.getAuthors()) {
        ViewEditBook.comboBoxAuthor.addItem(author.toString());
    }
}

// coloca os autores no "comboBoxPublisher" na tela de edição de livro
public static void updateEditComboPublisher() {
    PublisherDAO dao = new PublisherDAO();
    ViewEditBook.comboBoxPublisher.removeAllItems();
    ViewEditBook.comboBoxPublisher.addItem("Selecione uma editora");
    for (Publisher publisher: dao.getPublishers()) {
        ViewEditBook.comboBoxPublisher.addItem(publisher.toString());
    }
}

// coloca os autores no "comboBoxAuthor" na tela de adição de livro
public static void updateAddComboAuthor() {
    AuthorDAO dao = new AuthorDAO();
    ViewAddBook.comboBoxAuthor.removeAllItems();
    ViewAddBook.comboBoxAuthor.addItem("Selecione um autor(a)");
    for (Author author: dao.getAuthors()) {

```

```

        ViewAddBook.comboBoxAuthor.addItem(author.toString());
    }
}

// coloca os autores no "comboBoxPublisher" na tela de adição de livro
public static void updateAddComboPublisher() {
    PublisherDAO dao = new PublisherDAO();
    ViewAddBook.comboBoxPublisher.removeAllItems();
    ViewAddBook.comboBoxPublisher.addItem("Selecione uma editora");
    for (Publisher publisher: dao.getPublishers()) {
        ViewAddBook.comboBoxPublisher.addItem(publisher.toString());
    }
}

// separa a string de nome+sobrenome do autor
public static String splitAuthor(String toSplit, Integer position) {
    String [] splitted = toSplit.split(" ");
    return splitted[position];
}

// para trazer os dados de editora para a tabela
public static void readTableAuthorGeneral(String data) {
    DefaultTableModel modelo = (DefaultTableModel) ViewAuthor.tableAuthor.getModel();
    modelo.setNumRows(0);
    AuthorDAO pdao = new AuthorDAO();

    for (Author author: pdao.getAuthorsGeneral(data)) {
        modelo.addRow(new Object[] {
            author.getId(),
            author.getName(),
            author.getFName()
        });
    }

    if (modelo.getRowCount() == 0) {
        try {
            JOptionPane.showInternalMessageDialog(null, "Autor não encontrado.", "Aviso",
JOptionPane.WARNING_MESSAGE );
            ViewGeneral.buttonClean.setEnabled(false);
        } catch (Exception ignored) {}
    }
}

// para trazer os dados de editora para a tabela
public static void readTablePublisherGeneral(String data) {
    DefaultTableModel modelo = (DefaultTableModel) ViewPublisher.tablePublisher.getModel();
    modelo.setNumRows(0);
    PublisherDAO pdao = new PublisherDAO();

    for (Publisher publisher: pdao.getPublishersGeneral(data)) {
        modelo.addRow(new Object[] {
            publisher.getId(),
            publisher.getName(),
            publisher.getUrl()
        });
    }
}

```

```

        if (modelo.getRowCount() == 0) {
            try {
                JOptionPane.showInternalMessageDialog(null, "Editora não encontrada.", "Aviso",
JOptionPane.WARNING_MESSAGE );
                ViewGeneral.buttonClean.setEnabled(false);
            } catch (Exception ignored) {}
        }
    }

    // para trazer os dados de editora para a tabela
    public static void readTableBookGeneral(String data) {
        DefaultTableModel modelo = (DefaultTableModel) ViewBook.tableBook.getModel();
        modelo.setNumRows(0);
        BookDAO pdao = new BookDAO();

        for (Book book: pdao.getBooksGeneral(data)) {
            modelo.addRow(new Object[] {
                book.getPrice(),
                book.getTitle(),
                book.getPublisherId(),
                book.getIsbn()
            });
        }

        if (modelo.getRowCount() == 0) {
            try {
                JOptionPane.showInternalMessageDialog(null, "Livro não encontrado.", "Aviso",
JOptionPane.WARNING_MESSAGE );
                ViewGeneral.buttonClean.setEnabled(false);
            } catch (Exception ignored) {}
        }
    }

    public static Boolean checkForSpaces(String nome, String snome){
        return (nome.indexOf(" ") != -1 || snome.indexOf(" ") != -1);
    }
}

```

**> Pacote model.bean:**

Author.java

```

package model.bean;

public class Author {
    private Integer id;
    private String name;
    private String fName;

    public Author() {

    }

    public Author(String name, String fName) {
        this.name = name;
        this.fName = fName;
    }
}

```

```

public Author(Integer id, String name, String fName) {
    this.id = id;
    this.name = name;
    this.fName = fName;
}

public Integer getId() {
    return this.id;
}

public String getName() {
    return this.name;
}
public void setName(String name) {
    this.name = name;
}
public String getFName() {
    return this.fName;
}
public void setFName(String fName) {
    this.fName = fName;
}
public void setId (Integer id) {
    this.id = id;
}
@Override
public String toString() {
    return getName() + " " + getFName(); //To change body of generated methods, choose
Tools | Templates.
}

}

```

### Book.java

```

package model.bean;

public class Book {
    private String title;
    private String isbn;
    private Integer publisherId;
    private Double price;

    // construtor
    public Book(String title, String isbn, Integer publisherId, Double price) {
        this.title = title;
        this.isbn = isbn;
        this.publisherId = publisherId;
        this.price = price;
    }
    // construtor
    public Book() {

    }

    // getter do título

```

```

public String getTitle() {
    return title;
}

// setter do título
public void setTitle(String title) {
    this.title = title;
}

// getter do ISBN
public String getIsbn() {
    return isbn;
}

// setter do ISBN
public void setIsbn(String isbn) {
    this.isbn = isbn;
}

// getter do id da editora
public int getPublisherId() {
    return publisherId;
}

// setter do id da editora
public void setPublisherId(Integer publisherId) {
    this.publisherId = publisherId;
}

// getter de preço
public double getPrice() {
    return price;
}

// setter de preço
public void setPrice(Double price) {
    this.price = price;
}
}

```

#### BookAuthor.java

```

package model.bean;

public class BookAuthor {
    private String isbn;
    private Integer authorId;
    private Integer seqNo;

    public BookAuthor(String isbn, Integer authorId, Integer seqNo) {
        this.isbn = isbn;
        this.authorId = authorId;
        this.seqNo = seqNo;
    }

    public String getIsbn() {

```



```

        return isbn;
    }

    public void setISBN(String isbn) {
        this.isbn = isbn;
    }

    public Integer getAuthorId() {
        return authorId;
    }

    public void setAuthorId(Integer authorId) {
        this.authorId = authorId;
    }

    public Integer getSeqNo() {
        return seqNo;
    }

    public void setSeqNo(Integer seqNo) {
        this.seqNo = seqNo;
    }
}

```

Publisher.java

```

package model.bean;

public class Publisher {
    private int id;
    private String name;
    private String url;

    public Publisher(String name, String url) {
        this.name = name;
        this.url = url;
    }

    public Publisher(int publisherId, String name, String url) {
        this.id = publisherId;
        this.name = name;
        this.url = url;
    }

    public Publisher() {

    }

    public int getId() {
        return id;
    }

    public void setId(int publisherId) {

```

```

        this.id = publisherId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    @Override
    public String toString() {
        return getName();
    }
}

```

**> Pacote model.dao:**

AuthorDAO.java

```

package model.dao;

import connection.DatabaseConstants;
import model.bean.Author;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class AuthorDAO {
    private static final String URL = DatabaseConstants.URL;
    private static final String USER = DatabaseConstants.USER;
    private static final String PASS = DatabaseConstants.PASS;

    // Obtem todos os autores do banco
    public static List<Author> getAuthors() {
        List<Author> authors = new ArrayList<Author>();
        try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
            String query = "SELECT * FROM authors";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);

```

```

        while(rs.next()) {
            Integer id = rs.getInt("author_id");
            String name = rs.getString("name");
            String fname = rs.getString("fname");
            Author author = new Author(id, name, fname);
            authors.add(author);
        }

        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
    return authors;
}

// Obtem todos os autores do banco com o nome especificado
public static List<Author> getAuthorsStr(String aName) {
    List<Author> authors = new ArrayList<Author>();
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM authors WHERE name LIKE '%" + aName + "%'";
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        while(rs.next()) {
            Integer id = rs.getInt("author_id");
            String name = rs.getString("name");
            String fname = rs.getString("fname");
            Author author = new Author(id, name, fname);
            authors.add(author);
        }

        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
    return authors;
}

// Obtem todos os autores do banco com o nome/id/sobrenome especificado
public static List<Author> getAuthorsGeneral(String data) {
    List<Author> authors = new ArrayList<Author>();
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM authors WHERE author_id = ? "
            + "OR name = ? "
            + "OR fname = ?;";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(2, "%" + data + "%");
        pstmt.setString(3, "%" + data + "%");
        try {
            pstmt.setInt(1, Integer.parseInt(data));
        } catch (NumberFormatException e) {
            pstmt.setInt(1, -1);
        }
    }
}

```

```

        ResultSet rs = pstmt.executeQuery();

        while(rs.next()) {
            Integer id = rs.getInt("author_id");
            String name = rs.getString("name");
            String fname = rs.getString("fname");
            Author author = new Author(id, name, fname);
            authors.add(author);
        }
        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
    return authors;
}

// Obtem um autor do banco com determinado id
public static Author getAuthor(Integer id) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM authors WHERE author_id = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setInt(1, id);
        ResultSet rs = pstmt.executeQuery();

        rs.next();
        String name = rs.getString("name");
        String fname = rs.getString("fname");

        return new Author(id, name, fname);
    } catch(SQLException e) {
        e.printStackTrace();
        return null;
    }
}

// Obtem o id do autor de acordo com o nome e sobrenome
public static Integer getAuthorId(String name, String fname) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT author_id FROM authors WHERE name = ? AND fname = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, name);
        pstmt.setString(2, fname);
        ResultSet rs = pstmt.executeQuery();

        rs.next();
        Integer id = rs.getInt("author_id");

        return id;
    } catch(SQLException e) {
        e.printStackTrace();
        return null;
    }
}
}

```

```

// Obtem o nome de um autor do banco com determinado isbn
public static String getAuthorName(String isbn) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT CONCAT (name,\" \",fname) FROM authors \n" +
            "INNER JOIN booksauthors\n" +
            "ON booksauthors.author_id = authors.author_id\n" +
            "WHERE isbn= ? ;";
        PreparedStatement pstm = con.prepareStatement(query);
        pstm.setString(1, isbn);
        ResultSet rs = pstm.executeQuery();

        rs.next();
        String name = rs.getString("CONCAT (name,\" \",fname)");

        return name;
    } catch(SQLException e) {
        e.printStackTrace();
        return null;
    }
}

// Adiciona um autor ao banco
public static void addAuthor(Author author) {
    try(Connection conn = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "INSERT INTO authors (name, fname) VALUES (?, ?)";
        PreparedStatement pstm = conn.prepareStatement(query);
        pstm.setString(1, author.getName());
        pstm.setString(2, author.getFName());
        pstm.executeUpdate();
        conn.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
}

// Atualiza um autor no banco de dados
public static void updateAuthor(Author author) {
    try(Connection conn = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "UPDATE authors SET name = ?, fname = ? WHERE author_id = ?";
        PreparedStatement pstm = conn.prepareStatement(query);
        pstm.setString(1, author.getName());
        pstm.setString(2, author.getFName());
        pstm.setInt(3, author.getId());
        pstm.executeUpdate();
        conn.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
}

// Delete um autor no banco de dados que tenha o id especificado
public static void deleteAuthor(Integer id) {

```

```

try (Connection conn = DriverManager.getConnection(URL, USER, PASS)) {
    String query = "DELETE FROM authors WHERE author_id = ?";
    BookAuthorDAO.deleteBookAuthor(id);
    PreparedStatement pstm = conn.prepareStatement(query);
    pstm.setInt(1, id);
    pstm.executeUpdate();
    conn.close();
} catch (SQLException e) {
    e.printStackTrace();
}
}

// Deleta um autor no banco de dados que tenha o id igual ao do autor especificado
public static void deleteAuthor(Author author) {
    try (Connection conn = DriverManager.getConnection(URL, USER, PASS)) {

        String query = "DELETE FROM authors WHERE author_id = ?";
        BookAuthorDAO.deleteBookAuthor(author.getId());
        PreparedStatement pstm = conn.prepareStatement(query);
        pstm.setInt(1, author.getId());
        pstm.executeUpdate();
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

#### BookAuthorDAO.java

```

package model.dao;

import connection.DatabaseConstants;
import model.bean.BookAuthor;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class BookAuthorDAO {
    private static final String URL = DatabaseConstants.URL;
    private static final String USER = DatabaseConstants.USER;
    private static final String PASS = DatabaseConstants.PASS;

    // Obtem todos autores dos livros
    public static List<BookAuthor> getBookAuthors() {
        List<BookAuthor> bookAuthors = new ArrayList<BookAuthor>();
        try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
            String query = "SELECT * FROM booksauthors";

```

```

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(query);

while(rs.next()) {
    String isbn = rs.getString("isbn");
    Integer authorId = rs.getInt("author_id");
    Integer seqNo = rs.getInt("seq_no");

    BookAuthor ba = new BookAuthor(isbn, authorId, seqNo);
    bookAuthors.add(ba);
}

con.close();
} catch(SQLException e) {
    e.printStackTrace();
}
return bookAuthors;
}

// Obtem um autor do livro com isbn
public static BookAuthor getBookAuthor(String isbn) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM booksauthors WHERE isbn = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, isbn);
        ResultSet rs = pstmt.executeQuery();

        rs.next();
        Integer authorId = rs.getInt("author_id");
        Integer seqNo = rs.getInt("seq_no");

        return new BookAuthor(isbn, authorId, seqNo);
    } catch(SQLException e) {
        e.printStackTrace();
        return null;
    }
}

public static Integer getSeq(String isbn) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT seq_no FROM booksauthors WHERE isbn= ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, isbn);
        ResultSet rs = pstmt.executeQuery();

        rs.next();
        Integer id = rs.getInt("seq_no");

        return id;
    } catch(SQLException e) {
        e.printStackTrace();
        return null;
    }
}

```

```
}
```

```
// Adiciona um autor do livro
```

```
public static void addBookAuthor(BookAuthor bookAuthor) {  
    try(Connection conn = DriverManager.getConnection(URL, USER, PASS)) {  
        String query = "INSERT INTO booksauthors VALUES (?, ?, ?)";  
        PreparedStatement pstmt = conn.prepareStatement(query);  
        pstmt.setString(1, bookAuthor.getIsbn());  
        pstmt.setInt(2, bookAuthor.getAuthorId());  
        pstmt.setInt(3, bookAuthor.getSeqNo());  
        pstmt.executeUpdate();  
        conn.close();  
    } catch(SQLException e) {  
        e.printStackTrace();  
    }  
}
```

```
// Atualiza um autor do livro
```

```
public static void updateBookAuthor(BookAuthor bookAuthor) {  
    try(Connection conn = DriverManager.getConnection(URL, USER, PASS)) {  
        String query = "UPDATE booksauthors SET isbn = ?, author_id = ?, seq_no = ?";  
        PreparedStatement pstmt = conn.prepareStatement(query);  
        pstmt.setString(1, bookAuthor.getIsbn());  
        pstmt.setInt(2, bookAuthor.getAuthorId());  
        pstmt.setInt(3, bookAuthor.getSeqNo());  
        pstmt.executeUpdate();  
        conn.close();  
    } catch(SQLException e) {  
        e.printStackTrace();  
    }  
}
```

```
// Deleta um autor do livro
```

```
public static void deleteBookAuthor(String isbn) {  
    try (Connection conn = DriverManager.getConnection(URL, USER, PASS)) {  
        String query = "DELETE FROM booksauthors WHERE isbn = ?";  
        PreparedStatement pstmt = conn.prepareStatement(query);  
        pstmt.setString(1, isbn);  
        pstmt.executeUpdate();  
        conn.close();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```

```
public static void deleteBookAuthor(Integer authorId) {  
    try (Connection conn = DriverManager.getConnection(URL, USER, PASS)) {  
        String query = "DELETE FROM booksauthors WHERE author_id = ?";  
        PreparedStatement pstmt = conn.prepareStatement(query);  
        pstmt.setInt(1, authorId);  
        pstmt.executeUpdate();  
        conn.close();  
    } catch (SQLException e) {
```



```

        e.printStackTrace();
    }
}
}

```

## BookDAO.java

```

package model.dao;

import connection.DatabaseConstants;
import model.bean.Book;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import model.bean.Publisher;

public class BookDAO {
    private static final String URL = DatabaseConstants.URL;
    private static final String USER = DatabaseConstants.USER;
    private static final String PASS = DatabaseConstants.PASS;

    // Obtem todos os livros
    public static List<Book> getBooks() {
        List<Book> books = new ArrayList<Book>();
        try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
            String query = "SELECT * FROM books";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            while(rs.next()) {
                String title = rs.getString("title");
                String isbn = rs.getString("isbn");
                Integer publisherId = rs.getInt("publisher_id");
                Double price = rs.getDouble("price");

                Book book = new Book(title, isbn, publisherId, price);
                books.add(book);
            }

            con.close();
        } catch(SQLException e) {
            e.printStackTrace();
        }

        return books;
    }

    // Obtem todos os livros com determinado titulo

```

```

public static List<Book> getBooksStr(String bName) {
    List<Book> books = new ArrayList<Book>();
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM books WHERE title LIKE '%" + bName + "%'";
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        while(rs.next()) {
            String title = rs.getString("title");
            String isbn = rs.getString("isbn");
            Integer publisherId = rs.getInt("publisher_id");
            Double price = rs.getDouble("price");

            Book book = new Book(title, isbn, publisherId, price);
            books.add(book);
        }

        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }

    return books;
}

```

```

// Obtem todos os livros com determinada editora
public static List<Book> getBooksWithPublisher(Publisher p) {
    List<Book> books = new ArrayList<Book>();
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM books WHERE publisher_id = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setInt(1, p.getId());

        ResultSet rs = pstmt.executeQuery();
        while(rs.next()) {
            String title = rs.getString("title");
            String isbn = rs.getString("isbn");
            Integer publisherId = rs.getInt("publisher_id");
            Double price = rs.getDouble("price");

            Book book = new Book(title, isbn, publisherId, price);
            books.add(book);
        }

        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }

    return books;
}

```

// Obtem todos os livros do banco com o título/isbn/editora/preço especificado

```

public static List<Book> getBooksGeneral(String data) {
    List<Book> books = new ArrayList<Book>();
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM books WHERE title LIKE ? "
            + "OR isbn = ? "
            + "OR publisher_id = ? "
            + "OR price = ?;";
        PreparedStatement pstm = con.prepareStatement(query);
        pstm.setString(1, "%" + data + "%");
        try {
            pstm.setInt(2, Integer.parseInt(data));
            pstm.setInt(3, Integer.parseInt(data));
            pstm.setInt(4, Integer.parseInt(data));
        } catch (NumberFormatException e) {
            pstm.setInt(2, -1);
            pstm.setInt(3, -1);
            pstm.setInt(4, -1);
        }
        ResultSet rs = pstm.executeQuery();

        while(rs.next()) {
            String title = rs.getString("title");
            String isbn = rs.getString("isbn");
            Integer publisherId = rs.getInt("publisher_id");
            Double price = rs.getDouble("price");

            Book book = new Book(title, isbn, publisherId, price);
            books.add(book);
        }
        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
    return books;
}

// Obtem determinado livro de acordo com a ISBN
public static Book getBook(String isbn) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM books WHERE isbn = ?";
        PreparedStatement pstm = con.prepareStatement(query);
        pstm.setString(1, isbn);
        ResultSet rs = pstm.executeQuery();

        rs.next();
        String title = rs.getString("title");
        Integer publisherId = rs.getInt("publisher_id");
        Double price = rs.getDouble("price");

        return new Book(title, isbn, publisherId, price);
    } catch(SQLException e) {
        e.printStackTrace();
        return null;
    }
}

```

```

    }
}

// Adiciona um livro novo
public static void addBook(Book book) {
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "INSERT INTO books VALUES (?, ?, ?, ?)";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, book.getTitle());
        pstmt.setString(2, book.getIsbn());
        pstmt.setInt(3, book.getPublisherId());
        pstmt.setDouble(4, book.getPrice());
        pstmt.executeUpdate();
        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
}

// Atualiza um livro
public static void updateBook(Book book) {
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "UPDATE books SET title = ?, price = ? WHERE isbn = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, book.getTitle());
        pstmt.setDouble(2, book.getPrice());
        pstmt.setString(3, book.getIsbn());
        pstmt.executeUpdate();
        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
}

// Deleta um livro
public static void deleteBook(Book book) {
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "DELETE FROM books WHERE isbn = ?";
        BookAuthorDAO.deleteBookAuthor(book.getIsbn());
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, book.getIsbn());
        pstmt.executeUpdate();
        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
}

// Deleta todos os livros que tenha determinada editora
public static void deleteBook(Publisher publisher) {
    List<Book> books = getBooksWithPublisher(publisher);

    for(Book b: books) {

```

```

        deleteBook(b);
    }
}

```

#### PublisherDAO.java

```

package model.dao;

import connection.DatabaseConstants;
import model.bean.Publisher;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.util.ArrayList;
import java.util.List;
import java.sql.ResultSet;

public class PublisherDAO {

    private static final String URL = DatabaseConstants.URL;
    private static final String USER = DatabaseConstants.USER;
    private static final String PASS = DatabaseConstants.PASS;

    // Obtem todas as editoras
    public static List<Publisher> getPublishers() {
        List<Publisher> publishers = new ArrayList<Publisher>();

        try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
            String query = "SELECT * FROM publishers";
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            while(rs.next()) {
                Integer id = rs.getInt("publisher_id");
                String name = rs.getString("name");
                String url = rs.getString("url");

                Publisher publisher = new Publisher(id, name, url);
                publishers.add(publisher);
            }
            con.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return publishers;
    }

    // Obtem uma editora com determinado id
    public static Publisher getPublisher(Integer id) {

```

```

try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
    String query = "SELECT * FROM publishers WHERE publisher_id = ?";
    PreparedStatement pstmt = con.prepareStatement(query);
    pstmt.setInt(1, id);
    ResultSet rs = pstmt.executeQuery();

    rs.next();
    String name = rs.getString("name");
    String url = rs.getString("url");

    return new Publisher(id, name, url);
} catch (SQLException e) {
    e.printStackTrace();
    return null;
}
}

// Obtem todas as editoras que possuem determinado nome
public static List<Publisher> getPublisherStr(String name) {
    List<Publisher> publishers = new ArrayList<Publisher>();

    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM publishers WHERE name LIKE '%" + name + "%'";
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        while(rs.next()) {
            Integer id = rs.getInt("publisher_id");
            String pName = rs.getString("name");
            String url = rs.getString("url");

            Publisher publisher = new Publisher(id, pName, url);
            publishers.add(publisher);
        }
        con.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return publishers;
}

// Obtem todos as editoras do banco com o id/nome/url especificado
public static List<Publisher> getPublishersGeneral(String data) {
    List<Publisher> publishers = new ArrayList<Publisher>();
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT * FROM publishers WHERE publisher_id = ? "
            + "OR name LIKE ? "
            + "OR url LIKE ?;";
        PreparedStatement pstmt = con.prepareStatement(query);
        try {
            pstmt.setInt(1, Integer.parseInt(data));
        } catch (NumberFormatException e) {
    
```

```

        pstmt.setInt(1, -1);
    }
    pstmt.setString(2, "%" + data + "%");
    pstmt.setString(3, "%" + data + "%");
    ResultSet rs = pstmt.executeQuery();

    while(rs.next()) {
        Integer id = rs.getInt("publisher_id");
        String name = rs.getString("name");
        String url = rs.getString("url");

        Publisher publisher = new Publisher(id, name, url);
        publishers.add(publisher);
    }
    con.close();
} catch(SQLException e) {
    e.printStackTrace();
}
return publishers;
}

public static Integer getPublisherId(String name) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT publisher_id FROM publishers WHERE name= ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, name);
        ResultSet rs = pstmt.executeQuery();

        rs.next();
        Integer id = rs.getInt("publisher_id");

        return id;
    } catch(SQLException e) {
        e.printStackTrace();
        return null;
    }
}

public static String getPublisherName(Integer id) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT name FROM publishers " +
            "INNER JOIN books " +
            "ON books.publisher_id = publishers.publisher_id " +
            "WHERE publishers.publisher_id = ? " +
            "LIMIT 1";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setInt(1, id);
        ResultSet rs = pstmt.executeQuery();

        rs.next();
        String name = rs.getString("name");
    }
}

```

```

        return name;
    } catch(SQLException e) {
        e.printStackTrace();
        return null;
    }
}

// Adiciona uma editora nova ao banco
public static void addPublisher(Publisher publisher) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "INSERT INTO publishers (name, url) values (?, ?)";
        PreparedStatement pstm = con.prepareStatement(query);
        pstm.setString(1, publisher.getName());
        pstm.setString(2, publisher.getUrl());
        pstm.executeUpdate();
        con.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Atualiza determinada editora no banco
public static void updatePublisher(Publisher publisher) {
    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "UPDATE publishers set name = ?, url = ? WHERE publisher_id = ?";
        PreparedStatement pstm = con.prepareStatement(query);
        pstm.setString(1, publisher.getName());
        pstm.setString(2, publisher.getUrl());
        pstm.setInt(3, publisher.getId());
        pstm.executeUpdate();
        con.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Deleta determinada editora no banco
public void deletePublisher(Publisher publisher) {
    try(Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "DELETE FROM publishers WHERE publisher_id = ?";
        BookDAO.deleteBook(publisher);
        PreparedStatement pstm = con.prepareStatement(query);
        pstm.setInt(1, publisher.getId());
        pstm.executeUpdate();
        con.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
}
}

```

ViewModelsDAO.java



```

package model.dao;

import model.view.SearchViewModel;
import connection.DatabaseConstants;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class ViewModelsDAO {
    private static final String URL = DatabaseConstants.URL;
    private static final String USER = DatabaseConstants.USER;
    private static final String PASS = DatabaseConstants.PASS;

    // Método que retorna os dados referente a tela de busca
    public static List<SearchViewModel> getSearchViewModels() {
        List<SearchViewModel> searchViewModels = new ArrayList<SearchViewModel>();

        try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
            String query = "SELECT books.price, books.title, CONCAT (authors.name,\"
\",authors.fname) AS author, publishers.name AS publisher, books.isbn\n" +
                "FROM books INNER JOIN publishers INNER JOIN booksauthors INNER JOIN
authors\n" +
                "ON publishers.publisher_id = books.publisher_id\n" +
                "AND booksauthors.isbn = books.isbn\n" +
                "AND booksauthors.author_id = authors.author_id\n";

            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            while(rs.next()) {
                Double price = rs.getDouble("price");
                String title = rs.getString("title");
                String author = rs.getString("author");
                String publisher = rs.getString("publisher");
                String isbn = rs.getString("isbn");

                SearchViewModel vm = new SearchViewModel(price, title, author, publisher, isbn);
                searchViewModels.add(vm);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return searchViewModels;
    }

    // Método que retorna os dados referente a tela de busca de acordo com o titulo buscado

```

```

public static List<SearchViewModel> getSearchByTitle(String bookTitle) {
    List<SearchViewModel> searchViewModels = new ArrayList<SearchViewModel>();

    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT books.price, books.title, CONCAT (authors.name,\"
\",authors.fname) AS author, publishers.name AS publisher, books.isbn\n" +
            "FROM books INNER JOIN publishers INNER JOIN booksauthors INNER JOIN
authors\n" +
            "ON publishers.publisher_id = books.publisher_id\n" +
            "AND booksauthors.isbn = books.isbn\n" +
            "AND booksauthors.author_id = authors.author_id\n" +
            "WHERE books.title LIKE ? ;";
        PreparedStatement stmt = con.prepareStatement(query);
        stmt.setString(1, "%" + bookTitle + "%");

        ResultSet rs = stmt.executeQuery();

        while(rs.next()) {
            Double price = rs.getDouble("price");
            String title = rs.getString("title");
            String author = rs.getString("author");
            String publisher = rs.getString("publisher");
            String isbn = rs.getString("isbn");

            SearchViewModel vm = new SearchViewModel(price, title, author, publisher, isbn);
            searchViewModels.add(vm);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return searchViewModels;
}

```

```

// Método que retorna os dados referente a tela de busca de acordo com o autor buscado
public static List<SearchViewModel> getSearchByAuthor(String authorName) {
    List<SearchViewModel> searchViewModels = new ArrayList<SearchViewModel>();

    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT books.price, books.title, CONCAT (authors.name,\"
\",authors.fname) AS author, publishers.name AS publisher, books.isbn\n" +
            "FROM books INNER JOIN publishers INNER JOIN booksauthors INNER JOIN
authors\n" +
            "ON publishers.publisher_id = books.publisher_id\n" +
            "AND booksauthors.isbn = books.isbn\n" +
            "AND booksauthors.author_id = authors.author_id\n" +
            "WHERE authors.name LIKE ? \n" +
            "OR authors.fname LIKE ? ;";
        PreparedStatement stmt = con.prepareStatement(query);
        stmt.setString(1, authorName + "%");
        stmt.setString(2, authorName + "%");

        ResultSet rs = stmt.executeQuery();
    }
}

```

```

        while(rs.next()) {
            Double price = rs.getDouble("price");
            String title = rs.getString("title");
            String author = rs.getString("author");
            String publisher = rs.getString("publisher");
            String isbn = rs.getString("isbn");

            SearchViewModel vm = new SearchViewModel(price, title, author, publisher, isbn);
            searchViewModels.add(vm);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return searchViewModels;
}

// Método que retorna os dados referente a tela de busca de acordo com a editora buscada
public static List<SearchViewModel> getSearchByPublisher(String publisherName) {
    List<SearchViewModel> searchViewModels = new ArrayList<SearchViewModel>();

    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT books.price, books.title, CONCAT (authors.name,\"
\",authors.fname) AS author, publishers.name AS publisher, books.isbn\n" +
            "FROM books INNER JOIN publishers INNER JOIN booksauthors INNER JOIN
authors\n" +
            "ON publishers.publisher_id = books.publisher_id\n" +
            "AND booksauthors.isbn = books.isbn\n" +
            "AND booksauthors.author_id = authors.author_id\n" +
            "WHERE publishers.name LIKE ? ";
        PreparedStatement stmt = con.prepareStatement(query);
        stmt.setString(1, "%" + publisherName + "%");

        ResultSet rs = stmt.executeQuery();

        while(rs.next()) {
            Double price = rs.getDouble("price");
            String title = rs.getString("title");
            String author = rs.getString("author");
            String publisher = rs.getString("publisher");
            String isbn = rs.getString("isbn");

            SearchViewModel vm = new SearchViewModel(price, title, author, publisher, isbn);
            searchViewModels.add(vm);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return searchViewModels;
}

```

```

// Método que retorna os dados referente a tela de busca de acordo com o isbn buscado
public static List<SearchViewModel> getSearchByIsbn(String bookIsbn) {
    List<SearchViewModel> searchViewModels = new ArrayList<SearchViewModel>();

    try (Connection con = DriverManager.getConnection(URL, USER, PASS)) {
        String query = "SELECT books.price, books.title, CONCAT (authors.name,\"
\",authors.fname) AS author, publishers.name AS publisher, books.isbn\n" +
            "FROM books INNER JOIN publishers INNER JOIN booksauthors INNER JOIN
authors\n" +
            "ON publishers.publisher_id = books.publisher_id\n" +
            "AND booksauthors.isbn = books.isbn\n" +
            "AND booksauthors.author_id = authors.author_id\n" +
            "WHERE books.isbn LIKE ? ;";
        PreparedStatement stmt = con.prepareStatement(query);
        stmt.setString(1, bookIsbn);

        ResultSet rs = stmt.executeQuery();

        while(rs.next()) {
            Double price = rs.getDouble("price");
            String title = rs.getString("title");
            String author = rs.getString("author");
            String publisher = rs.getString("publisher");
            String isbn = rs.getString("isbn");

            SearchViewModel vm = new SearchViewModel(price, title, author, publisher, isbn);
            searchViewModels.add(vm);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return searchViewModels;
}
}

```

**> Pacote model.view:**

SearchViewModel.java

```

package model.view;

// Classe que representa os dados da tela de busca
public class SearchViewModel {
    private Double price;
    private String title;
    private String author;
    private String publisher;
    private String isbn;

    public SearchViewModel(Double price, String title, String author, String publisher, String isbn) {
        this.price = price;
        this.title = title;
    }
}

```

```

        this.author = author;
        this.publisher = publisher;
        this.isbn = isbn;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }

    public String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }
}

```

**> Pacote view:**

ViewAbout.java

```

package view;

import java.awt.Dimension;
import java.beans.PropertyVetoException;

```

```

import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;

public class ViewAbout extends javax.swing.JInternalFrame {

    public ViewAbout() {
        initComponents();
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewLivrariaAmazonia.aboutIsOpen = false;
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        labelAbout = new javax.swing.JLabel();
        buttonClose = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();

        setClosable(true);
        setIconifiable(true);
        setTitle("Sobre");

        labelAbout.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
        labelAbout.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        labelAbout.setText("Livraria Amazônia:");

        buttonClose.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
exit.png"))); // NOI18N
        buttonClose.setText("Fechar");
        buttonClose.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonCloseActionPerformed(evt);
            }
        });

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Sistema criado pelos integrantes:");
        jLabel1.setToolTipText("APS 4º Sem. Unip Swift");
        jLabel1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        jLabel1.setFocusable(false);
        jLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("Gabriel Menezes de Antonio - F13GJI6");
    }

```



```

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
        .addGap(0, 26, Short.MAX_VALUE)
        .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 350,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(23, 23, 23))
        .addGroup(layout.createSequentialGroup()
        .addGap(48, 48, 48)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 300,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 300,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 300,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 300,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(labelAbout)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(buttonClose)
        .addContainerGap())
    );

    pack();
} // </editor-fold>

// fecha a janela atual
private void buttonCloseActionPerformed(java.awt.event.ActionEvent evt) {

```



```

    try {
        this.setClosed(true);
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
    ViewLivrariaAmazonia.aboutIsOpen = false;
}

// coloca a janela atual na posição central do programa
public void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

// Variables declaration - do not modify
private javax.swing.JButton buttonClose;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel labelAbout;
// End of variables declaration
}

```

#### ViewAddAuthor.java

```

package view;

import controller.ControllerView;
import model.dao.AuthorDAO;
import model.bean.Author;
import java.awt.Dimension;
import java.awt.event.KeyEvent;
import java.beans.PropertyVetoException;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;

public class ViewAddAuthor extends javax.swing.JInternalFrame {

    public ViewAddAuthor() {
        initComponents();
        buttonAdd.setEnabled(false);
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewAuthor.addAuthorIsOpen = false;
            }
        });
    }
}

```

```

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    panelAdd = new javax.swing.JPanel();
    buttonAdd = new javax.swing.JButton();
    buttonCancel = new javax.swing.JButton();
    labelName = new javax.swing.JLabel();
    labelLastName = new javax.swing.JLabel();
    textName = new javax.swing.JTextField();
    textLastName = new javax.swing.JTextField();

    setClosable(true);
    setTitle("Adicionar Autor");
    setMinimumSize(new java.awt.Dimension(421, 192));

    panelAdd.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0,
0)));

    buttonAdd.setText("Adicionar");
    buttonAdd.setMaximumSize(null);
    buttonAdd.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonAdd.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonAdd.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonAddActionPerformed(evt);
        }
    });

    buttonCancel.setText("Cancelar");
    buttonCancel.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonCancel.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonCancel.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonCancelActionPerformed(evt);
        }
    });

    labelName.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
    labelName.setText("Nome:");

    labelLastName.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
    labelLastName.setText("Sobrenome:");

    textName.setToolTipText("Digite o nome do autor");
    textName.addKeyListener(new java.awt.event.KeyAdapter() {
        public void keyReleased(java.awt.event.KeyEvent evt) {
            textNameKeyReleased(evt);
        }
        public void keyTyped(java.awt.event.KeyEvent evt) {
            textNameKeyTyped(evt);
        }
    });
}

```



```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 16,
Short.MAX_VALUE)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(buttonAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap()
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(panelAdd, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addContainerGap()
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(panelAdd, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

// quando é clicado em "Cancelar" , fecha a janela interna "Adicionar livro"
private void buttonCancelActionPerformed(java.awt.event.ActionEvent evt) {
    closeWindow();
}

// quando uma tecla é solta no "textTitle", chama o método verifyText()
private void textNameKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// quando uma tecla é solta no "textAuthor", chama o método verifyText()
private void textLastNameKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// adiciona um novo autor no banco de dados
private void buttonAddActionPerformed(java.awt.event.ActionEvent evt) {
    String name = textName.getText();
    String lastName = textLastName.getText();
    if(!ControllerView.checkForSpaces(name, lastName)){

```

```

        Author author = new Author(name, lastName);
        AuthorDAO.addAuthor(author);

        try {
            ControllerView.readTableAuthor();
            ViewAuthor.buttonEdit.setEnabled(false);
            ViewAuthor.buttonDelete.setEnabled(false);
            ViewAddBook.comboBoxAuthor.addItem(author.toString());
        } catch (NullPointerException ignored) {}

        Object[] options = { "Sim", "Não" };
        Icon figura = new ImageIcon
(getToolkit().createImage(getClass().getResource("../images/icon-done.png")));
        int option = JOptionPane.showOptionDialog(null, "Autor(a) adicionado.\nGostaria de
adicionar mais?", "Adicionar autor(a)", JOptionPane.DEFAULT_OPTION,
JOptionPane.PLAIN_MESSAGE, figura, options, options[1]);

        if (option == 1 || option == -1) {
            closeWindow();
        } else if (option == 0) {
            textName.setText("");
            textLastName.setText("");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Nomes e sobrenomes não podem conter
espaços.");
    }

}

// limita a quantidade de caracteres em "Nome" para 25 e não permite espaço
private void textNameKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textName.getText() + c).length() > 25) || c == KeyEvent.VK_SPACE) {
        evt.consume();
    }
}

// limita a quantidade de caracteres em "Sobrenome" para 25 e não permite espaço
private void textLastNameKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textLastName.getText() + c).length() > 25) || c == KeyEvent.VK_SPACE) {
        evt.consume();
    }
}

// fecha a janela atual
private void closeWindow() {
    try {
        this.setClosed(true);
        ViewAuthor.addAuthorIsOpen = false;
    } catch (PropertyVetoException ex) {

```

```

        System.err.println("Closing Exception");
    }
}

// define a posição da janela interna no centro do programa
protected void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

/* verifica se existe texto nos campos "Título", "Autor", "Editora", "ISBN" e "Preço"
   caso todos tenham texto: habilita o botão "Adicionar"
   caso não: desabilita o botão "Adicionar" */
private void verifyText() {
    String textT = textName.getText();
    String textA = textLastName.getText();

    if (textT.isBlank() || textA.isBlank()) {
        buttonAdd.setEnabled(false);
    } else {
        buttonAdd.setEnabled(true);
    }
}

// Variables declaration - do not modify
private javax.swing.JButton buttonAdd;
private javax.swing.JButton buttonCancel;
private javax.swing.JLabel labelLastName;
private javax.swing.JLabel labelName;
private javax.swing.JPanel panelAdd;
private javax.swing.JTextField textLastName;
private javax.swing.JTextField textName;
// End of variables declaration
}

```

#### ViewAddBook.java

```

package view;

import controller.ControllerView;
import java.awt.Dimension;
import java.awt.event.KeyEvent;
import java.beans.PropertyVetoException;
import java.math.RoundingMode;
import java.text.DecimalFormat;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;
import model.bean.*;
import model.dao.*;

```

```

public class ViewAddBook extends javax.swing.JInternalFrame {

    protected ViewAddBook() {
        initComponents();
        buttonAdd.setEnabled(false);
        ControllerView.updateAddComboAuthor();
        ControllerView.updateAddComboPublisher();
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewBook.addBooksOpen = false;
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        panelAdd = new javax.swing.JPanel();
        buttonAdd = new javax.swing.JButton();
        buttonCancel = new javax.swing.JButton();
        labelTitle = new javax.swing.JLabel();
        labelAuthor = new javax.swing.JLabel();
        labelPublisher = new javax.swing.JLabel();
        labelPrice = new javax.swing.JLabel();
        labelISBN = new javax.swing.JLabel();
        textTitle = new javax.swing.JTextField();
        textPrice = new javax.swing.JTextField();
        textISBN = new javax.swing.JTextField();
        comboBoxAuthor = new javax.swing.JComboBox<>();
        comboBoxPublisher = new javax.swing.JComboBox<>();
        buttonAddAuthor = new javax.swing.JButton();
        buttonAddPublisher = new javax.swing.JButton();
        labelSequence = new javax.swing.JLabel();
        textSequence = new javax.swing.JTextField();

        setClosable(true);
        setTitle("Adicionar Livro");

        panelAdd.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

        buttonAdd.setText("Adicionar");
        buttonAdd.setMaximumSize(null);
        buttonAdd.setMinimumSize(new java.awt.Dimension(134, 22));
        buttonAdd.setPreferredSize(new java.awt.Dimension(134, 22));
        buttonAdd.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonAddActionPerformed(evt);
            }
        });

        buttonCancel.setText("Cancelar");
    }
}

```

```

buttonCancel.setMinimumSize(new java.awt.Dimension(134, 22));
buttonCancel.setPreferredSize(new java.awt.Dimension(134, 22));
buttonCancel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonCancelActionPerformed(evt);
    }
});

labelTitle.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelTitle.setText("Título:");

labelAuthor.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelAuthor.setText("Autor:");

labelPublisher.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelPublisher.setText("Editora:");

labelPrice.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelPrice.setText("Preço:");

labelIsbn.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelIsbn.setText("ISBN:");

textTitle.setToolTipText("Digite o título do livro");
textTitle.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textTitleKeyReleased(evt);
    }
    public void keyTyped(java.awt.event.KeyEvent evt) {
        textTitleKeyTyped(evt);
    }
});

textPrice.setToolTipText("Digite o valor do livro. Exemplo: 32.99");
textPrice.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textPriceKeyReleased(evt);
    }
    public void keyTyped(java.awt.event.KeyEvent evt) {
        textPriceKeyTyped(evt);
    }
});

textIsbn.setToolTipText("Digite o ISBN do livro (ATENÇÃO: para editar posteriormente,
deverá excluir o livro)");
textIsbn.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textIsbnKeyReleased(evt);
    }
    public void keyTyped(java.awt.event.KeyEvent evt) {
        textIsbnKeyTyped(evt);
    }
});

```



```

comboBoxAuthor.setToolTipText("Selecione o autor(a)");
comboBoxAuthor.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(java.awt.event.FocusEvent evt) {
        comboBoxAuthorFocusGained(evt);
    }
});
comboBoxAuthor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxAuthorActionPerformed(evt);
    }
});

comboBoxPublisher.setToolTipText("Selecione a editora");
comboBoxPublisher.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(java.awt.event.FocusEvent evt) {
        comboBoxPublisherFocusGained(evt);
    }
});
comboBoxPublisher.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxPublisherActionPerformed(evt);
    }
});

buttonAddAuthor.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
buttonAddAuthor.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-add-book.png"))); // NOI18N
buttonAddAuthor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonAddAuthorActionPerformed(evt);
    }
});

buttonAddPublisher.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
buttonAddPublisher.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-add-book.png"))); // NOI18N
buttonAddPublisher.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonAddPublisherActionPerformed(evt);
    }
});

labelSequence.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelSequence.setText("Seq. nº:");

textSequence.setToolTipText("Digite a sequência do livro, caso não seja de uma coleção,
digite 1");
textSequence.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textSequenceKeyReleased(evt);
    }
    public void keyTyped(java.awt.event.KeyEvent evt) {

```

```

        textSequenceKeyTyped(evt);
    }
});

javax.swing.GroupLayout panelAddLayout = new javax.swing.GroupLayout(panelAdd);
panelAdd.setLayout(panelAddLayout);
panelAddLayout.setHorizontalGroup(
    panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelAddLayout.createSequentialGroup()
            .addComponent(labelTitle)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(textTitle, javax.swing.GroupLayout.PREFERRED_SIZE, 331,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(0, 0, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelAddLayout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(labelSequence)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(textSequence, javax.swing.GroupLayout.PREFERRED_SIZE,
331, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(panelAddLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(buttonAdd, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(buttonCancel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGroup(panelAddLayout.createSequentialGroup()
            .addContainerGap()
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelAddLayout.createSequentialGroup()
                    .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                        .addGroup(panelAddLayout.createSequentialGroup()
                            .addGap(8, 8, 8)
                            .addComponent(labelAuthor)
                            .addComponent(labelPublisher)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        )
                        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                            .addComponent(comboBoxAuthor, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(comboBoxPublisher, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    )
                )
            )

```

```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
    )
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING, false)
            .addComponent(buttonAddAuthor,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(buttonAddPublisher)))
        .addGroup(panelAddLayout.createSequentialGroup())
        .addGap(9, 9, 9)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.TRAILING)
            .addComponent(labelIsbn)
            .addComponent(labelPrice))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
    )
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addComponent(textIsbn)
            .addComponent(textPrice))))))
        .addContainerGap())
    );
    panelAddLayout.setVerticalGroup(
        panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelAddLayout.createSequentialGroup()
            .addContainerGap()
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(textTitle)
                .addComponent(labelTitle))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(textSequence)
                .addComponent(labelSequence))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(labelAuthor)
                .addComponent(comboBoxAuthor, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonAddAuthor))
            .addGap(13, 13, 13)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(labelPublisher)
                .addComponent(comboBoxPublisher, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonAddPublisher))
            .addGap(12, 12, 12)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(textIsbn)

```

```

        .addComponent(labelIsbn))
        .addGap(13, 13, 13)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(textPrice)
        .addComponent(labelPrice))
        .addGap(18, 18, 18)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(buttonAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap())
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(panelAdd, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addContainerGap())
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(panelAdd, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

// quando é clicado em "Cancelar", fecha a janela interna "Adicionar livro"
private void buttonCancelActionPerformed(java.awt.event.ActionEvent evt) {
    closeWindow();
}

// quando uma tecla é solta no "textTitle", chama o método verifyText()
private void textTitleKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// quando uma tecla é solta no "textPrice", chama o método verifyText()
private void textPriceKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

```

```

// adiciona um novo livro e autor de livro no banco de dados
private void buttonAddActionPerformed(java.awt.event.ActionEvent evt) {
    DecimalFormat df = new DecimalFormat("0.00");
    df.setRoundingMode(RoundingMode.HALF_UP);

    // pegando titulo, isbn, sequencia, preço e id(editora)
    String title = textTitle.getText();
    String isbn = textIsbn.getText();
    Integer seqNo = Integer.parseInt(textSequence.getText());
    Double price = Double.parseDouble(textPrice.getText());
    df.format(price);
    Integer publisherId = PublisherDAO.getPublisherId((String)
comboBoxPublisher.getModel().getSelectedItem());

    // pegando o id do autor, a partir do nome e sobrenome
    String name = ControllerView.splitAuthor((String)
comboBoxAuthor.getModel().getSelectedItem(), 0);
    String fname = ControllerView.splitAuthor((String)
comboBoxAuthor.getModel().getSelectedItem(), 1);
    Integer authorId = AuthorDAO.getAuthorId(name, fname);

    // adicionando o livro e o autor do livro
    Book book = new Book(title, isbn, publisherId, price);
    BookDAO.addBook(book);
    BookAuthor bookAuthor = new BookAuthor(isbn, authorId, seqNo);
    BookAuthorDAO.addBookAuthor(bookAuthor);

    // atualizando a tabela da tela de livros
    ControllerView.readTableBook();

    // chamando a janela de opções para saber se o usuário quer adicionar mais livros
    Object[] options = { "Sim", "Não" };
    Icon figura = new ImageIcon
(getToolkit().createImage(getClass().getResource("../images/icon-done.png")));
    int option = JOptionPane.showOptionDialog(null, "Livro adicionado.\nGostaria de adicionar
mais?", "Adicionar livro", JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE,
figura, options, options[1]);

    if (option == 1 || option == -1) {
        closeWindow();
    } else if (option == 0) {
        textTitle.setText("");
        textIsbn.setText("");
        textPrice.setText("");
        textSequence.setText("");
        comboBoxAuthor.setSelectedItem(0);
        comboBoxPublisher.setSelectedItem(0);
    }
}

// chama a janela interna para adicionar um novo autor, limitando para 1 janela dessa aberta
private void buttonAddAuthorActionPerformed(java.awt.event.ActionEvent evt) {
    if (!ViewAuthor.addAuthorIsOpen) {

```

```

        ViewAddAuthor viewAddAuthor = new ViewAddAuthor();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewAddAuthor);
        viewAddAuthor.setVisible(true);
        viewAddAuthor.setPositionCenter();
        ViewAuthor.addAuthorIsOpen = true;
    }
}

// chama a janela interna para adicionar uma nova editora, limitando para 1 janela dessa
aberta
private void buttonAddPublisherActionPerformed(java.awt.event.ActionEvent evt) {
    if (!ViewPublisher.addPublisherIsOpen) {
        ViewAddPublisher viewAddPublisher = new ViewAddPublisher();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewAddPublisher);
        viewAddPublisher.setVisible(true);
        viewAddPublisher.setPositionCenter();
        ViewPublisher.addPublisherIsOpen = true;
    }
}

// quando uma tecla é solta no "Seq. N", chama o método verifyText()
private void textSequenceKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// quando um autor(a) é selecionado, chama o método verifyText()
private void comboBoxAuthorActionPerformed(java.awt.event.ActionEvent evt) {
    verifyText();
}

// quando uma editora é selecionado, chama o método verifyText()
private void comboBoxPublisherActionPerformed(java.awt.event.ActionEvent evt) {
    verifyText();
}

// limita a quantidade de caracteres em "ISBN" para 13 e proíbe espaço
private void textIsbnKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textIsbn.getText() + c).length() > 13) || c == KeyEvent.VK_SPACE) {
        evt.consume();
    }
}

// quando uma tecla é solta no "ISBN", chama o método verifyText()
private void textIsbnKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// limita a quantidade de caracteres em "Título" para 60
private void textTitleKeyTyped(java.awt.event.KeyEvent evt) {
    if ((textTitle.getText() + evt.getKeyChar()).length() > 60) {
        evt.consume();
    }
}

```

```

}

// limita a quantidade de caracteres em "Seq No." para 11, proíbe espaço e aceita só números
private void textSequenceKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textSequence.getText() + c).length() > 11) || c == KeyEvent.VK_SPACE
|| !Character.isDigit(c)) {
        evt.consume();
    }
}

// limita a quantidade de caracteres em "Preço" para 11, para 11, proíbe espaço e só aceita
números decimais
private void textPriceKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    String text = textPrice.getText();
    if (((text + c).length() > 11) || c == KeyEvent.VK_SPACE || Character.isLetter(c)) {
        evt.consume();
    } else {
        try {
            Double.parseDouble(text + c);
        } catch (NumberFormatException e) {
            evt.consume();
        }
    }
}

private void comboBoxPublisherFocusGained(java.awt.event.FocusEvent evt) {
    ControllerView.updateAddComboPublisher();
}

private void comboBoxAuthorFocusGained(java.awt.event.FocusEvent evt) {
    ControllerView.updateAddComboAuthor();
}

// fecha a janela atual
private void closeWindow() {
    try {
        this.setClosed(true);
        ViewBook.addBookIsOpen = false;
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
}

// define a posição da janela interna no centro do programa
protected void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

/* verifica se existe texto nos campos "Título", "Autor", "Editora", "ISBN" e "Preço"
caso todos tenham texto: habilita o botão "Adicionar"

```

```

        caso não: desabilita o botão "Adicionar" */
private void verifyText() {
    String textT = textTitle.getText();
    String textI = textIsbn.getText();
    String textPr = textPrice.getText();
    String textS = textSequence.getText();
    int comboA = comboBoxAuthor.getSelectedIndex();
    int comboP = comboBoxPublisher.getSelectedIndex();

    if (textT.isBlank() || textI.isBlank() || textPr.isBlank() || textS.isBlank() || comboA == 0 ||
    comboP == 0) {
        buttonAdd.setEnabled(false);
    } else {
        buttonAdd.setEnabled(true);
    }
}

// Variables declaration - do not modify
private javax.swing.JButton buttonAdd;
private javax.swing.JButton buttonAddAuthor;
private javax.swing.JButton buttonAddPublisher;
private javax.swing.JButton buttonCancel;
public static javax.swing.JComboBox<Object> comboBoxAuthor;
public static javax.swing.JComboBox<Object> comboBoxPublisher;
private javax.swing.JLabel labelAuthor;
private javax.swing.JLabel labelIsbn;
private javax.swing.JLabel labelPrice;
private javax.swing.JLabel labelPublisher;
private javax.swing.JLabel labelSequence;
private javax.swing.JLabel labelTitle;
private javax.swing.JPanel panelAdd;
private javax.swing.JTextField textIsbn;
private javax.swing.JTextField textPrice;
private javax.swing.JTextField textSequence;
private javax.swing.JTextField textTitle;
// End of variables declaration
}

```

#### ViewAddPublisher.java

```

package view;

import controller.ControllerView;
import model.bean.Publisher;
import model.dao.PublisherDAO;
import java.awt.Dimension;
import java.awt.event.KeyEvent;
import java.beans.PropertyVetoException;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;

```



```

public class ViewAddPublisher extends javax.swing.JInternalFrame {

    protected ViewAddPublisher() {
        initComponents();
        buttonAdd.setEnabled(false);
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewPublisher.addPublisherIsOpen = false;
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        panelAdd = new javax.swing.JPanel();
        buttonAdd = new javax.swing.JButton();
        buttonCancel = new javax.swing.JButton();
        labelName = new javax.swing.JLabel();
        labelURL = new javax.swing.JLabel();
        textName = new javax.swing.JTextField();
        textUrl = new javax.swing.JTextField();

        setClosable(true);
        setTitle("Adicionar Editora");

        panelAdd.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

        buttonAdd.setText("Adicionar");
        buttonAdd.setMaximumSize(null);
        buttonAdd.setMinimumSize(new java.awt.Dimension(134, 22));
        buttonAdd.setPreferredSize(new java.awt.Dimension(134, 22));
        buttonAdd.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonAddActionPerformed(evt);
            }
        });

        buttonCancel.setText("Cancelar");
        buttonCancel.setMinimumSize(new java.awt.Dimension(134, 22));
        buttonCancel.setPreferredSize(new java.awt.Dimension(134, 22));
        buttonCancel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonCancelActionPerformed(evt);
            }
        });

        labelName.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
        labelName.setText("Nome:");
    }
}

```

```

labelURL.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelURL.setText("URL:");

textName.setToolTipText("Digite o nome da editora");
textName.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textNameKeyReleased(evt);
    }
    public void keyTyped(java.awt.event.KeyEvent evt) {
        textNameKeyTyped(evt);
    }
});

textUrl.setToolTipText("Digite a URL(website) da editora");
textUrl.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textUrlKeyReleased(evt);
    }
    public void keyTyped(java.awt.event.KeyEvent evt) {
        textUrlKeyTyped(evt);
    }
});

javax.swing.GroupLayout panelAddLayout = new javax.swing.GroupLayout(panelAdd);
panelAdd.setLayout(panelAddLayout);
panelAddLayout.setHorizontalGroup(
    panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelAddLayout.createSequentialGroup()
            .add(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(buttonAdd, javax.swing.GroupLayout.DEFAULT_SIZE, 188, Short.MAX_VALUE)
                .add(buttonCancel, javax.swing.GroupLayout.DEFAULT_SIZE, 189, Short.MAX_VALUE))
            .add(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(labelName, javax.swing.GroupLayout.Alignment.TRAILING)
                .add(labelURL, javax.swing.GroupLayout.Alignment.TRAILING)
                .add(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .add(textName)
                    .add(textUrl))))
            .addContainerGap(8, 8, 8)
        );
panelAddLayout.setVerticalGroup(
    panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .add(panelAddLayout.createSequentialGroup()
            .add(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(buttonAdd)
                .add(buttonCancel))
            .add(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(labelName)
                .add(labelURL)
                .add(textName)
                .add(textUrl))
            .addContainerGap(8, 8, 8)
        );

```

```

        .addContainerGap()
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
            .addComponent(textName)
            .addComponent(labelName))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
            .addComponent(textUrl)
            .addComponent(labelURL))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 15,
Short.MAX_VALUE)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
            .addComponent(buttonAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap()
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(panelAdd, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap())
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(panelAdd, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

// quando é clicado em "Cancelar" , fecha a janela interna "Adicionar livro"
private void buttonCancelActionPerformed(java.awt.event.ActionEvent evt) {
    closeWindow();
}

// quando uma tecla é solta no "textTitle", chama o método verifyText()
private void textNameKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

```

```

// quando uma tecla é solta no "textAuthor", chama o método verifyText()
private void textUrlKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// quando é clicado, adiciona uma editora no banco de dados
private void buttonAddActionPerformed(java.awt.event.ActionEvent evt) {
    String name = textName.getText();
    String url = textUrl.getText();

    Publisher publisher = new Publisher(name, url);
    PublisherDAO.addPublisher(publisher);

    try {
        ControllerView.readTablePublisher();
        ViewPublisher.buttonDelete.setEnabled(false);
        ViewPublisher.buttonEdit.setEnabled(false);
    } catch (NullPointerException ignored) {}

    Object[] options = { "Sim", "Não" };
    Icon figura = new ImageIcon
(getToolkit().createImage(getClass().getResource("../images/icon-done.png")));
    int option = JOptionPane.showOptionDialog(null, "Editora adicionada.\nGostaria de
adicionar mais?", "Adicionar editora", JOptionPane.DEFAULT_OPTION,
JOptionPane.PLAIN_MESSAGE, figura, options, options[1]);

    if (option == 1 || option == -1) {
        closeWindow();
    } else if (option == 0) {
        textName.setText("");
        textUrl.setText("");
    }
}

// limita a quantidade de caracteres em "Nome" para 30
private void textNameKeyTyped(java.awt.event.KeyEvent evt) {
    if ((textName.getText() + evt.getKeyChar()).length() > 30) {
        evt.consume();
    }
}

// limita a quantidade de caracteres em "URL" para 80 e proíbe espaço
private void textUrlKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textUrl.getText() + c).length() > 80) || c == KeyEvent.VK_SPACE) {
        evt.consume();
    }
}

// fecha a janela atual
private void closeWindow() {
    try {
        this.setClosed(true);
    }
}

```

```

        ViewPublisher.addPublisherIsOpen = false;
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
}

// define a posição da janela interna no centro do programa
protected void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

/* verifica se existe texto nos campos "Título", "Autor", "Editora", "ISBN" e "Preço"
   caso todos tenham texto: habilita o botão "Adicionar"
   caso não: desabilita o botão "Adicionar" */
private void verifyText() {
    String textN = textName.getText();
    String textU = textUrl.getText();

    if (textN.isBlank() || textU.isBlank()) {
        buttonAdd.setEnabled(false);
    } else {
        buttonAdd.setEnabled(true);
    }
}

// Variables declaration - do not modify
private javax.swing.JButton buttonAdd;
private javax.swing.JButton buttonCancel;
private javax.swing.JLabel labelName;
private javax.swing.JLabel labelURL;
private javax.swing.JPanel panelAdd;
private javax.swing.JTextField textName;
private javax.swing.JTextField textUrl;
// End of variables declaration
}

```

#### ViewAuthor.java

```

package view;

import controller.ControllerView;
import java.awt.Dimension;
import java.beans.PropertyVetoException;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;
import javax.swing.table.DefaultTableModel;
import model.dao.AuthorDAO;
import view.ViewAddAuthor;

public class ViewAuthor extends javax.swing.JInternalFrame {

    static boolean addAuthorIsOpen = false;

```

```

static boolean editAuthorIsOpen = false;

protected ViewAuthor() {
    initComponents();
    buttonEdit.setEnabled(false);
    buttonDelete.setEnabled(false);
    buttonSearch.setEnabled(false);
    buttonShowAll.setEnabled(false);
    ControllerView.readTableAuthor();
    tableAuthor.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
    addInternalFrameListener(new InternalFrameAdapter(){
        public void internalFrameClosing(InternalFrameEvent e) {
            ViewLivrariaAmazonia.authorIsOpen = false;
        }
    });
}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    panelEdit = new javax.swing.JPanel();
    buttonAdd = new javax.swing.JButton();
    buttonFechar = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    tableAuthor = new javax.swing.JTable();
    buttonDelete = new javax.swing.JButton();
    buttonEdit = new javax.swing.JButton();
    textSearch = new javax.swing.JTextField();
    buttonSearch = new javax.swing.JButton();
    buttonClear = new javax.swing.JButton();
    buttonShowAll = new javax.swing.JButton();

    setClosable(true);
    setIconifiable(true);
    setMaximizable(true);
    setResizable(true);
    setTitle("Autores");

    panelEdit.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

    buttonAdd.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-add-book.png"))); // NOI18N
    buttonAdd.setText("Adicionar");
    buttonAdd.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonAdd.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonAdd.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonAddActionPerformed(evt);
        }
    });
}

```

```

        buttonFechar.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
exit.png"))); // NOI18N
        buttonFechar.setText("Fechar");
        buttonFechar.setMinimumSize(new java.awt.Dimension(134, 22));
        buttonFechar.setPreferredSize(new java.awt.Dimension(134, 22));
        buttonFechar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonFecharActionPerformed(evt);
            }
        });

        tableAuthor.setAutoCreateRowSorter(true);
        tableAuthor.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

                },
            new String [] {
                "ID", "Nome", "Sobrenome"
            }
        )
        {
            public boolean isCellEditable(int row, int column) {
                return false;
            }
        }
    );
    tableAuthor.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            tableAuthorMouseClicked(evt);
        }
    });
    jScrollPane1.setViewportViewView(tableAuthor);

    buttonDelete.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
delete.png"))); // NOI18N
    buttonDelete.setText("Excluir");
    buttonDelete.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonDelete.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonDelete.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonDeleteActionPerformed(evt);
        }
    });

    buttonEdit.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
pencil.png"))); // NOI18N
    buttonEdit.setText("Editar");
    buttonEdit.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonEdit.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonEdit.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonEditActionPerformed(evt);
        }
    });

```

```

});

textSearch.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textSearchKeyReleased(evt);
    }
});

buttonSearch.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
search.png"))); // NOI18N
buttonSearch.setText("Buscar");
buttonSearch.setPreferredSize(new java.awt.Dimension(119, 22));
buttonSearch.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonSearchActionPerformed(evt);
    }
});

buttonClear.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
clear.png"))); // NOI18N
buttonClear.setText("Limpar");
buttonClear.setPreferredSize(new java.awt.Dimension(119, 22));
buttonClear.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonClearActionPerformed(evt);
    }
});

buttonShowAll.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
show.png"))); // NOI18N
buttonShowAll.setText("Mostrar tudo");
buttonShowAll.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonShowAllActionPerformed(evt);
    }
});

javax.swing.GroupLayout panelEditLayout = new javax.swing.GroupLayout(panelEdit);
panelEdit.setLayout(panelEditLayout);
panelEditLayout.setHorizontalGroup(
    panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelEditLayout.createSequentialGroup()
            .add(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(panelEditLayout.createSequentialGroup()
                    .addContainerGap()
                    .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 632,
Short.MAX_VALUE)
                        .addGroup(panelEditLayout.createSequentialGroup()
                            .addComponent(buttonAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(buttonEdit, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE)

```



```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonDelete, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonFechar, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE))
        .addGroup(panelEditLayout.createSequentialGroup())
        .addComponent(textSearch)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonSearch, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonClear, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonShowAll)))
        .addContainerGap()
    );
    panelEditLayout.setVerticalGroup(
        panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelEditLayout.createSequentialGroup())
        .addContainerGap()
        .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
            .addComponent(buttonSearch, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(textSearch, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(buttonClear, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonShowAll)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 266,
Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                .addComponent(buttonAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonFechar, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonDelete, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonEdit, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap()
            );

```

```
panelEditLayout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
{buttonAdd, buttonClear, buttonDelete, buttonEdit, buttonFechar, buttonSearch, buttonShowAll,
textSearch});
```

```
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(panelEdit, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap())
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(panelEdit, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap())
);

pack();
} // </editor-fold>
```

```
// quando é clicado em "Cancelar", fecha a janela interna "Autores"
private void buttonFecharActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        this.setClosed(true);
        ViewLivrariaAmazonia.authorIsOpen = false;
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
}
```

```
// quando um item da tabela é selecionado, ativa os botões "buttonEdit" e "buttonDelete"
private void tableAuthorMouseClicked(java.awt.event.MouseEvent evt) {
    buttonEdit.setEnabled(true);
    buttonDelete.setEnabled(true);
}
```

// quando é clicado em "Adicionar" chama a view "ViewAddAuthor", limitando para 1 janela dessa aberta

```
private void buttonAddActionPerformed(java.awt.event.ActionEvent evt) {
    if (!addAuthorIsOpen) {
        ViewAddAuthor viewAddAuthor = new ViewAddAuthor();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewAddAuthor);
        viewAddAuthor.setVisible(true);
        viewAddAuthor.setPositionCenter();
        addAuthorIsOpen = true;
    }
}
```

```

// quando é clicado em "Excluir" exclui o autor do banco de dados e atualiza a tabela
private void buttonDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    Integer id = ((int) tableAuthor.getValueAt(tableAuthor.getSelectedRow(), 0));
    AuthorDAO.deleteAuthor(id);
    buttonDelete.setEnabled(false);
    buttonEdit.setEnabled(false);
    ControllerView.readTableAuthor();
}

/* quando é clicado em "editar" chama a internalFrame "viewAuthor" já com as
   informações do autor selecionado, limitando para 1 janela dessa aberta */
private void buttonEditActionPerformed(java.awt.event.ActionEvent evt) {
    if (!editAuthorIsOpen) {
        ViewEditAuthor viewEditAuthor = new ViewEditAuthor();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewEditAuthor);
        viewEditAuthor.setVisible(true);
        viewEditAuthor.setPositionCenter();
        viewEditAuthor.txtId.setEnabled(false);

        String name = ((String) tableAuthor.getValueAt(tableAuthor.getSelectedRow(), 1));
        viewEditAuthor.txtName.setText(name);

        String lastName = ((String) tableAuthor.getValueAt(tableAuthor.getSelectedRow(), 2));
        viewEditAuthor.txtLastName.setText(lastName);

        Integer id = ((int) tableAuthor.getValueAt(tableAuthor.getSelectedRow(), 0));
        viewEditAuthor.txtId.setText(id.toString());
        editAuthorIsOpen = true;
    }
}

// realiza a pesquisa de autor
private void buttonSearchActionPerformed(java.awt.event.ActionEvent evt) {
    String search = textSearch.getText();
    ControllerView.readTableAuthorGeneral(search);
    buttonClear.setEnabled(true);
    buttonShowAll.setEnabled(true);
}

// limpa a busca
private void buttonClearActionPerformed(java.awt.event.ActionEvent evt) {
    buttonClear.setEnabled(false);
    buttonSearch.setEnabled(false);
    buttonShowAll.setEnabled(true);
    DefaultTableModel modelo = (DefaultTableModel) tableAuthor.getModel();
    modelo.setNumRows(0);
    textSearch.setText("");
}

/* quando uma tecla é solta dentro do campo de busca
   verifica se existe texto no campo de busca
   caso sim: habilita o botão "Pesquisar"

```

```

    caso não: desabilita o botão "Pesquisar" */
private void textSearchKeyReleased(java.awt.event.KeyEvent evt) {
    String text = textSearch.getText();

    if (text.isBlank()) {
        buttonSearch.setEnabled(false);
    } else {
        buttonSearch.setEnabled(true);
    }
}

// mostra todos os autores para o usuário
private void buttonShowAllActionPerformed(java.awt.event.ActionEvent evt) {
    ControllerView.readTableAuthor();
    buttonClear.setEnabled(true);
    buttonShowAll.setEnabled(false);
}

// define a posição da janela interna no centro do programa
protected void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

// Variables declaration - do not modify
private javax.swing.JButton buttonAdd;
private javax.swing.JButton buttonClear;
protected static javax.swing.JButton buttonDelete;
protected static javax.swing.JButton buttonEdit;
private javax.swing.JButton buttonFechar;
private javax.swing.JButton buttonSearch;
private javax.swing.JButton buttonShowAll;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JPanel panelEdit;
public static javax.swing.JTable tableAuthor;
private javax.swing.JTextField textSearch;
// End of variables declaration
}

```

#### ViewBook.java

```

package view;

import controller.ControllerView;
import java.awt.Dimension;
import java.beans.PropertyVetoException;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;
import javax.swing.table.DefaultTableModel;
import model.bean.*;
import model.dao.*;

public class ViewBook extends javax.swing.JInternalFrame {

```

```

static boolean addBooksOpen = false;
static boolean editBooksOpen = false;

protected ViewBook() {
    initComponents();
    buttonEdit.setEnabled(false);
    buttonDelete.setEnabled(false);
    buttonShowAll.setEnabled(false);
    buttonSearch.setEnabled(false);
    ControllerView.readTableBook();
    tableBook.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
    addInternalFrameListener(new InternalFrameAdapter(){
        public void internalFrameClosing(InternalFrameEvent e) {
            ViewLivrariaAmazonia.booksOpen = false;
        }
    });
}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    panelEdit = new javax.swing.JPanel();
    jScrollPane1 = new javax.swing.JScrollPane();
    tableBook = new javax.swing.JTable();
    buttonAdd = new javax.swing.JButton();
    buttonEdit = new javax.swing.JButton();
    buttonDelete = new javax.swing.JButton();
    buttonCancel = new javax.swing.JButton();
    textSearch = new javax.swing.JTextField();
    buttonSearch = new javax.swing.JButton();
    buttonClear = new javax.swing.JButton();
    buttonShowAll = new javax.swing.JButton();

    setClosable(true);
    setIconifiable(true);
    setMaximizable(true);
    setResizable(true);
    setTitle("Livros");

    panelEdit.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0,
0)));

    tableBook.setAutoCreateRowSorter(true);
    tableBook.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {

        },
        new String [] {
            "Título", "ISBN", "Preço", "Editora"
        }
    )
}

```

```

        {
            public boolean isCellEditable(int row, int column) {
                return false;
            }
        }
    };
    tableBook.setCellSelectionEnabled(true);
    tableBook.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            tableBookMouseClicked(evt);
        }
    });
    jScrollPane1.setViewportViewView(tableBook);

    buttonAdd.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-add-book.png"))); // NOI18N
    buttonAdd.setText("Adicionar");
    buttonAdd.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonAdd.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonAdd.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonAddActionPerformed(evt);
        }
    });

    buttonEdit.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-pencil.png"))); // NOI18N
    buttonEdit.setText("Editar");
    buttonEdit.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonEdit.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonEdit.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonEditActionPerformed(evt);
        }
    });

    buttonDelete.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-delete.png"))); // NOI18N
    buttonDelete.setText("Excluir");
    buttonDelete.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonDelete.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonDelete.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonDeleteActionPerformed(evt);
        }
    });

    buttonCancel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-exit.png"))); // NOI18N
    buttonCancel.setText("Fechar");
    buttonCancel.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonCancel.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonCancel.addActionListener(new java.awt.event.ActionListener() {

```

```

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonCancelActionPerformed(evt);
        }
    });

    textSearch.addKeyListener(new java.awt.event.KeyAdapter() {
        public void keyReleased(java.awt.event.KeyEvent evt) {
            textSearchKeyReleased(evt);
        }
    });

    buttonSearch.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
search.png"))); // NOI18N
    buttonSearch.setText("Buscar");
    buttonSearch.setPreferredSize(new java.awt.Dimension(119, 22));
    buttonSearch.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonSearchActionPerformed(evt);
        }
    });

    buttonClear.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
clear.png"))); // NOI18N
    buttonClear.setText("Limpar");
    buttonClear.setPreferredSize(new java.awt.Dimension(119, 22));
    buttonClear.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonClearActionPerformed(evt);
        }
    });

    buttonShowAll.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
show.png"))); // NOI18N
    buttonShowAll.setText("Mostrar tudo");
    buttonShowAll.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonShowAllActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout panelEditLayout = new javax.swing.GroupLayout(panelEdit);
    panelEdit.setLayout(panelEditLayout);
    panelEditLayout.setHorizontalGroup(
        panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelEditLayout.createSequentialGroup()
                .add(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .add(panelEditLayout.createSequentialGroup()
                        .addGroup(panelEditLayout.createSequentialGroup()
                            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 632,
Short.MAX_VALUE)
                            .addGroup(panelEditLayout.createSequentialGroup()
                                .addComponent(buttonAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE)

```

```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonEdit, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonDelete, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelEditLayout.createSequentialGroup())
        .addComponent(textSearch)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonSearch, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonClear, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonShowAll)))
        .addContainerGap())
    );
    panelEditLayout.setVerticalGroup(
        panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelEditLayout.createSequentialGroup())
        .addContainerGap()
        .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
            .addComponent(buttonSearch, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(textSearch, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonClear, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonShowAll)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 265,
Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                .addComponent(buttonAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonDelete, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonEdit, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap())
        );

```





```

}

// quando é clicado em "Excluir" exclui o livro do banco de dados e atualiza a tabela
private void buttonDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    Book book = new Book();
    BookDAO bookDAO = new BookDAO();

    book.setIsbn((String) tableBook.getValueAt(tableBook.getSelectedRow(), 3));
    bookDAO.deleteBook(book);
    buttonEdit.setEnabled(false);
    buttonDelete.setEnabled(false);
    ControllerView.readTableBook();
}

// quando é clicado em "Editar" abre a janela interna de edição já com os dados do livro
selecionado, limitando para 1 janela dessa aberta
private void buttonEditActionPerformed(java.awt.event.ActionEvent evt) {
    if (!editBookIsOpen) {
        ViewEditBook viewEditBook = new ViewEditBook();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewEditBook);
        viewEditBook.setVisible(true);
        viewEditBook.setPositionCenter();

        Double price = ((double) tableBook.getValueAt(tableBook.getSelectedRow(), 0));
        viewEditBook.textPrice.setText(price.toString());

        String title = ((String) tableBook.getValueAt(tableBook.getSelectedRow(), 1));
        viewEditBook.textTitle.setText(title);

        String isbn = ((String) tableBook.getValueAt(tableBook.getSelectedRow(), 3));
        viewEditBook.textIsbn.setText(isbn);

        Integer sequence = BookAuthorDAO.getSeq(isbn);
        viewEditBook.textSequence.setText(sequence.toString());

        String author = AuthorDAO.getAuthorName(isbn);
        viewEditBook.comboBoxAuthor.setSelectedItem(author);

        String publisher = PublisherDAO.getPublisherName(((Integer)
tableBook.getValueAt(tableBook.getSelectedRow(), 2)));
        viewEditBook.comboBoxPublisher.setSelectedItem(publisher);
        editBookIsOpen = true;
    }
}

private void textSearchKeyReleased(java.awt.event.KeyEvent evt) {
    String text = textSearch.getText();

    if (text.isBlank()) {
        buttonSearch.setEnabled(false);
    } else {
        buttonSearch.setEnabled(true);
    }
}

```

```

    }

    // realiza a pesquisa de livro
    private void buttonSearchActionPerformed(java.awt.event.ActionEvent evt) {
        String search = textSearch.getText();
        ControllerView.readTableBookGeneral(search);
        buttonClear.setEnabled(true);
        buttonShowAll.setEnabled(true);
    }

    private void buttonClearActionPerformed(java.awt.event.ActionEvent evt) {
        buttonClear.setEnabled(false);
        buttonSearch.setEnabled(false);
        buttonShowAll.setEnabled(true);
        DefaultTableModel modelo = (DefaultTableModel) tableBook.getModel();
        modelo.setNumRows(0);
        textSearch.setText("");
    }

    private void buttonShowAllActionPerformed(java.awt.event.ActionEvent evt) {
        ControllerView.readTableBook();
        buttonClear.setEnabled(true);
        buttonShowAll.setEnabled(false);
    }

    // define a posição da janela interna no centro do programa
    protected void setPositionCenter() {
        Dimension d = this.getDesktopPane().getSize();
        this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
    }

    // Variables declaration - do not modify
    private javax.swing.JButton buttonAdd;
    private javax.swing.JButton buttonCancel;
    private javax.swing.JButton buttonClear;
    private javax.swing.JButton buttonDelete;
    private javax.swing.JButton buttonEdit;
    private javax.swing.JButton buttonSearch;
    private javax.swing.JButton buttonShowAll;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JPanel panelEdit;
    public static javax.swing.JTable tableBook;
    private javax.swing.JTextField textSearch;
    // End of variables declaration
}

```

ViewEditAuthor.java

```

package view;

import controller.ControllerView;
import model.dao.AuthorDAO;
import model.bean.Author;

```

```

import java.awt.Dimension;
import java.awt.event.KeyEvent;
import java.beans.PropertyVetoException;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;

public class ViewEditAuthor extends javax.swing.JInternalFrame {

    public ViewEditAuthor() {
        initComponents();
        buttonSave.setEnabled(false);
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewAuthor.editAuthorIsOpen = false;
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        panelAdd = new javax.swing.JPanel();
        buttonSave = new javax.swing.JButton();
        buttonCancel = new javax.swing.JButton();
        labelName = new javax.swing.JLabel();
        labelLastName = new javax.swing.JLabel();
        textName = new javax.swing.JTextField();
        textLastName = new javax.swing.JTextField();
        labelId = new javax.swing.JLabel();
        textId = new javax.swing.JTextField();

        setClosable(true);
        setTitle("Editor Autor");
        setMinimumSize(new java.awt.Dimension(421, 192));

        panelAdd.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

        buttonSave.setText("Salvar");
        buttonSave.setMinimumSize(new java.awt.Dimension(134, 22));
        buttonSave.setPreferredSize(new java.awt.Dimension(134, 22));
        buttonSave.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonSaveActionPerformed(evt);
            }
        });

        buttonCancel.setText("Cancelar");
        buttonCancel.setMinimumSize(new java.awt.Dimension(134, 22));

```



```

        .addGap(8, 8, 8)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(labelName)
            .addComponent(labelLastName)
            .addComponent(labelId))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(textId)
            .addComponent(textName)
            .addComponent(textLastName))))
        .addContainerGap()
    );
    panelAddLayout.setVerticalGroup(
        panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelAddLayout.createSequentialGroup()
            .addContainerGap()
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(textId)
                .addComponent(labelId))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(textName)
                .addComponent(labelName))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(textLastName)
                .addComponent(labelLastName))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 15,
                Short.MAX_VALUE)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(buttonSave, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap())
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(panelAdd, javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap())
    );

```

```

        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(panelAdd, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap(9, Short.MAX_VALUE))
                );

        pack();
    } // </editor-fold>

    // quando é clicado em "Cancelar", fecha a janela interna "Adicionar livro"
    private void buttonCancelActionPerformed(java.awt.event.ActionEvent evt) {
        closeWindow();
    }

    // quando uma tecla é solta no "textTitle", chama o método verifyText()
    private void textNameKeyReleased(java.awt.event.KeyEvent evt) {
        verifyText();
    }

    // quando uma tecla é solta no "textAuthor", chama o método verifyText()
    private void textLastNameKeyReleased(java.awt.event.KeyEvent evt) {
        verifyText();
    }

    // atualiza o autor selecionado no banco de dados
    private void buttonSaveActionPerformed(java.awt.event.ActionEvent evt) {
        String name = textName.getText();
        String lastName = textLastName.getText();
        if(!ControllerView.checkForSpaces(name, lastName)){
            Integer id = ((int)
ViewAuthor.tableAuthor.getValueAt(ViewAuthor.tableAuthor.getSelectedRow(), 0));

            Author author = new Author(id, name, lastName);
            AuthorDAO.updateAuthor(author);

            Object[] options = { "Ok" };
            Icon figura = new ImageIcon
(getToolkit().createImage(getClass().getResource("../images/icon-done.png")));
            JOptionPane.showOptionDialog(null, "Autor(a) atualizado!", "Editar autor(a)",
JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE, figura, options, options[0]);
            closeWindow();
            ControllerView.readTableAuthor();
            ViewAuthor.buttonEdit.setEnabled(false);
            ViewAuthor.buttonDelete.setEnabled(false);
        } else {
            JOptionPane.showMessageDialog(null, "Nomes e sobrenomes não podem conter
espaços.");
        }
    }
}

```

```

// limita a quantidade de caracteres em "Nome" para 25 e não permite espaço
private void textNameKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textName.getText() + c).length() > 25) || c == KeyEvent.VK_SPACE) {
        evt.consume();
    }
}

// limita a quantidade de caracteres em "Sobrenome" para 25 e não permite espaço
private void textLastNameKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textLastName.getText() + c).length() > 25) || c == KeyEvent.VK_SPACE) {
        evt.consume();
    }
}

// fecha a janela atual
private void closeWindow() {
    try {
        this.setClosed(true);
        ViewAuthor.editAuthorIsOpen = false;
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
}

// define a posição da janela interna no centro do programa
protected void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

/* verifica se existe texto nos campos "Título", "Autor", "Editora", "ISBN" e "Preço"
   caso todos tenham texto: habilita o botão "Adicionar"
   caso não: desabilita o botão "Adicionar" */
private void verifyText() {
    String textT = textName.getText();
    String textA = textLastName.getText();

    if (textT.isBlank() || textA.isBlank()) {
        buttonSave.setEnabled(false);
    } else {
        buttonSave.setEnabled(true);
    }
}

// Variables declaration - do not modify
private javax.swing.JButton buttonCancel;
private javax.swing.JButton buttonSave;
private javax.swing.JLabel labelId;
private javax.swing.JLabel labelLastName;
private javax.swing.JLabel labelName;
private javax.swing.JPanel panelAdd;

```



```

protected javax.swing.JTextField textId;
protected javax.swing.JTextField textLastName;
protected javax.swing.JTextField textName;
// End of variables declaration
}

```

#### ViewEditBook.java

```

package view;

import controller.ControllerView;
import java.awt.Dimension;
import java.awt.event.KeyEvent;
import java.beans.PropertyVetoException;
import java.math.RoundingMode;
import java.text.DecimalFormat;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;
import model.bean.*;
import model.dao.*;

public class ViewEditBook extends javax.swing.JInternalFrame {

    protected ViewEditBook() {
        initComponents();
        buttonSave.setEnabled(false);
        textIsbn.setEnabled(false);
        ControllerView.updateEditComboAuthor();
        ControllerView.updateEditComboPublisher();
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewBook.editBooksOpen = false;
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        panelAdd = new javax.swing.JPanel();
        buttonSave = new javax.swing.JButton();
        buttonCancel = new javax.swing.JButton();
        labelTitle = new javax.swing.JLabel();
        labelAuthor = new javax.swing.JLabel();
        labelPublisher = new javax.swing.JLabel();
        labelPrice = new javax.swing.JLabel();
        labelIsbn = new javax.swing.JLabel();
        textTitle = new javax.swing.JTextField();
        textPrice = new javax.swing.JTextField();
    }

```

```

textIsbn = new javax.swing.JTextField();
comboBoxAuthor = new javax.swing.JComboBox<>();
comboBoxPublisher = new javax.swing.JComboBox<>();
buttonAddAuthor = new javax.swing.JButton();
buttonAddPublisher = new javax.swing.JButton();
labelSequence = new javax.swing.JLabel();
textSequence = new javax.swing.JTextField();

setClosable(true);
setTitle("Editor Livros");

panelAdd.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0,
0)));

buttonSave.setText("Salvar");
buttonSave.setMinimumSize(new java.awt.Dimension(134, 22));
buttonSave.setPreferredSize(new java.awt.Dimension(134, 22));
buttonSave.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonSaveActionPerformed(evt);
    }
});

buttonCancel.setText("Cancelar");
buttonCancel.setMinimumSize(new java.awt.Dimension(134, 22));
buttonCancel.setPreferredSize(new java.awt.Dimension(134, 22));
buttonCancel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonCancelActionPerformed(evt);
    }
});

labelTitle.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelTitle.setText("Título:");

labelAuthor.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelAuthor.setText("Autor:");

labelPublisher.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelPublisher.setText("Editora:");

labelPrice.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelPrice.setText("Preço:");

labelIsbn.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelIsbn.setText("ISBN:");

textTitle.setToolTipText("Digite o título do livro");
textTitle.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textTitleKeyReleased(evt);
    }
    public void keyTyped(java.awt.event.KeyEvent evt) {

```

```

        textTitleKeyTyped(evt);
    }
});

textPrice.setToolTipText("Digite o valor do livro. Exemplo: 32.99");
textPrice.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textPriceKeyReleased(evt);
    }
    public void keyTyped(java.awt.event.KeyEvent evt) {
        textPriceKeyTyped(evt);
    }
});

textIsbn.setToolTipText("ISBN do livro (não editável, para alterar é necessário excluir e
adicionar novamente)");

comboBoxAuthor.setToolTipText("Selecione o autor(a)");
comboBoxAuthor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxAuthorActionPerformed(evt);
    }
});

comboBoxPublisher.setToolTipText("Selecione a editora");
comboBoxPublisher.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        comboBoxPublisherActionPerformed(evt);
    }
});

buttonAddAuthor.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
buttonAddAuthor.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-add-book.png"))); // NOI18N
buttonAddAuthor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonAddAuthorActionPerformed(evt);
    }
});

buttonAddPublisher.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
buttonAddPublisher.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-add-book.png"))); // NOI18N
buttonAddPublisher.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonAddPublisherActionPerformed(evt);
    }
});

labelSequence.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
labelSequence.setText("Seq. n°:");

```

[illegible]

```

        .addComponent(buttonAddAuthor,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(buttonAddPublisher,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
        .addGap(0, 0, Short.MAX_VALUE))
        .addGroup(panelAddLayout.createSequentialGroup())
        .addGap(9, 9, 9)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING)
        .addComponent(labelIsbn)
        .addComponent(labelPrice))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
        .addComponent(textIsbn, javax.swing.GroupLayout.PREFERRED_SIZE, 1,
Short.MAX_VALUE)
        .addComponent(textPrice)))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelAddLayout.createSequentialGroup())
        .addGap(0, 0, Short.MAX_VALUE)
        .addComponent(labelSequence)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(textSequence, javax.swing.GroupLayout.PREFERRED_SIZE,
331, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap())
    );
    panelAddLayout.setVerticalGroup(
        panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelAddLayout.createSequentialGroup())
        .addContainerGap()
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(textTitle)
        .addComponent(labelTitle))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(textSequence)
        .addComponent(labelSequence))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(labelAuthor)
        .addComponent(comboBoxAuthor, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonAddAuthor))
        .addGap(13, 13, 13)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(labelPublisher)

```

```

        .addComponent(comboBoxPublisher, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonAddPublisher))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(textIsbn)
        .addComponent(labelIsbn))
        .addGap(13, 13, 13)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(textPrice)
        .addComponent(labelPrice))
        .addGap(13, 13, 13)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(buttonSave, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap())
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(panelAdd, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addContainerGap())
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(panelAdd, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

// quando é clicado em "Cancelar" , fecha a janela interna "Adicionar livro"
private void buttonCancelActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        this.setClosed(true);
        ViewBook.editBooksOpen = false;
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
}

```

```

    }

    // quando uma tecla é solta no "textTitle", chama o método verifyText()
    private void textTitleKeyReleased(java.awt.event.KeyEvent evt) {
        verifyText();
    }

    // quando uma tecla é solta no "textPrice", chama o método verifyText()
    private void textPriceKeyReleased(java.awt.event.KeyEvent evt) {
        verifyText();
    }

    // edita um livro no banco de dados
    private void buttonSaveActionPerformed(java.awt.event.ActionEvent evt) {
        String title = textTitle.getText();
        String isbn = textIsbn.getText();
        Integer publisherId = PublisherDAO.getPublisherId((String)
        comboBoxPublisher.getModel().getSelectedItem());
        DecimalFormat df = new DecimalFormat("0.00");
        df.setRoundingMode(RoundingMode.HALF_UP);
        Double price = Double.parseDouble(textPrice.getText());
        df.format(price);

        Book book = new Book(title, isbn, publisherId, price);
        BookDAO.updateBook(book);

        Object[] options = { "Ok" };
        Icon figura = new ImageIcon
        (getToolkit().createImage(getClass().getResource("../images/icon-done.png")));
        int option = JOptionPane.showOptionDialog(null, "Livro atualizado!", "Editar livro",
        JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE, figura, options, options[0]);

        if (option == -1) {
            try {
                this.setClosed(true);
            } catch (PropertyVetoException ex) {
                System.err.println("Closing Exception");
            }
        }
    }

    // chama a janela de adicionar autor, limitando para 1 janela dessa aberta
    private void buttonAddAuthorActionPerformed(java.awt.event.ActionEvent evt) {
        if (!ViewAuthor.addAuthorIsOpen) {
            ViewAddAuthor viewAddAuthor = new ViewAddAuthor();
            ViewLivrariaAmazonia.desktopAmazonia.add(viewAddAuthor);
            viewAddAuthor.setVisible(true);
            viewAddAuthor.setPositionCenter();
            ViewAuthor.addAuthorIsOpen = true;
        }
    }

    // chama a janela de adicionar editora, limitando para 1 janela dessa aberta

```

```

private void buttonAddPublisherActionPerformed(java.awt.event.ActionEvent evt) {
    if (!ViewPublisher.addPublisherIsOpen) {
        ViewAddPublisher viewAddPublisher = new ViewAddPublisher();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewAddPublisher);
        viewAddPublisher.setVisible(true);
        viewAddPublisher.setPositionCenter();
        ViewPublisher.addPublisherIsOpen = true;
    }
}

// quando uma tecla é solta no "Seq. N", chama o método verifyText()
private void textSequenceKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// quando é clicado em "Autor(a)", chama o método verifyText()
private void comboBoxAuthorActionPerformed(java.awt.event.ActionEvent evt) {
    verifyText();
}

// quando é clicado em "Editora", chama o método verifyText()
private void comboBoxPublisherActionPerformed(java.awt.event.ActionEvent evt) {
    verifyText();
}

// limita a quantidade de caracteres em "Título" para 60
private void textTitleKeyTyped(java.awt.event.KeyEvent evt) {
    if ((textTitle.getText() + evt.getKeyChar()).length() > 60) {
        evt.consume();
    }
}

// limita a quantidade de caracteres em "Seq No." para 11, proíbe espaço e aceita só números
private void textSequenceKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textSequence.getText() + c).length() > 11) || c == KeyEvent.VK_SPACE
|| !Character.isDigit(c)) {
        evt.consume();
    }
}

// limita a quantidade de caracteres em "Preço" para 11, para 11, proíbe espaço e só aceita
números decimais
private void textPriceKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    String text = textPrice.getText();
    if (((text + c).length() > 11) || c == KeyEvent.VK_SPACE || Character.isLetter(c)) {
        evt.consume();
    } else {
        try {
            Double.parseDouble(text + c);
        } catch (NumberFormatException e) {
            evt.consume();
        }
    }
}

```



```

    }
}

// define a posição da janela interna no centro do programa
protected void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

/* verifica se existe texto nos campos "Título", "Autor", "Editora", "ISBN" e "Preço"
   caso todos tenham texto: habilita o botão "Adicionar"
   caso não: desabilita o botão "Adicionar" */
private void verifyText() {
    String textT = textTitle.getText();
    String textI = textIsbn.getText();
    String textPr = textPrice.getText();
    String textS = textSequence.getText();
    int comboA = comboBoxAuthor.getSelectedIndex();
    int comboP = comboBoxPublisher.getSelectedIndex();

    if (textT.isBlank() || textI.isBlank() || textPr.isBlank() || textS.isBlank() || comboA == 0 ||
    comboP == 0) {
        buttonSave.setEnabled(false);
    } else {
        buttonSave.setEnabled(true);
    }
}

// Variables declaration - do not modify
private javax.swing.JButton buttonAddAuthor;
private javax.swing.JButton buttonAddPublisher;
private javax.swing.JButton buttonCancel;
private javax.swing.JButton buttonSave;
public static javax.swing.JComboBox<Object> comboBoxAuthor;
public static javax.swing.JComboBox<Object> comboBoxPublisher;
private javax.swing.JLabel labelAuthor;
private javax.swing.JLabel labelIsbn;
private javax.swing.JLabel labelPrice;
private javax.swing.JLabel labelPublisher;
private javax.swing.JLabel labelSequence;
private javax.swing.JLabel labelTitle;
private javax.swing.JPanel panelAdd;
protected javax.swing.JTextField textIsbn;
protected javax.swing.JTextField textPrice;
protected javax.swing.JTextField textSequence;
protected javax.swing.JTextField textTitle;
// End of variables declaration
}

```

ViewEditPublisher.java

package view;

```

import controller.ControllerView;
import java.awt.Dimension;
import java.awt.event.KeyEvent;
import java.beans.PropertyVetoException;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;
import model.bean.Publisher;
import model.dao.PublisherDAO;

public class ViewEditPublisher extends javax.swing.JInternalFrame {

    public ViewEditPublisher() {
        initComponents();
        buttonSave.setEnabled(false);
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewPublisher.editPublisherIsOpen = false;
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        panelAdd = new javax.swing.JPanel();
        buttonSave = new javax.swing.JButton();
        buttonCancel = new javax.swing.JButton();
        labelName = new javax.swing.JLabel();
        labelUrl = new javax.swing.JLabel();
        textName = new javax.swing.JTextField();
        textUrl = new javax.swing.JTextField();
        labelId = new javax.swing.JLabel();
        textId = new javax.swing.JTextField();

        setClosable(true);
        setTitle("Editor Editora");
        setMinimumSize(new java.awt.Dimension(421, 192));

        panelAdd.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

        buttonSave.setText("Salvar");
        buttonSave.setMinimumSize(new java.awt.Dimension(134, 22));
        buttonSave.setPreferredSize(new java.awt.Dimension(134, 22));
        buttonSave.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonSaveActionPerformed(evt);
            }
        }

```



```

        .addComponent(buttonSave, javax.swing.GroupLayout.DEFAULT_SIZE, 188,
Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonCancel, javax.swing.GroupLayout.DEFAULT_SIZE, 189,
Short.MAX_VALUE))
        .addGroup(panelAddLayout.createSequentialGroup())
        .addGap(8, 8, 8)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING)
            .addComponent(labelName)
            .addComponent(labelUrl)
            .addComponent(labelId))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING)
            .addComponent(textId)
            .addComponent(textName)
            .addComponent(textUrl))))
        .addContainerGap()
    );
    panelAddLayout.setVerticalGroup(
        panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelAddLayout.createSequentialGroup()
            .addContainerGap()
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                .addComponent(textId)
                .addComponent(labelId))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                .addComponent(textName)
                .addComponent(labelName))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                .addComponent(textUrl)
                .addComponent(labelUrl))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 15,
Short.MAX_VALUE)
            .addGroup(panelAddLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)
                .addComponent(buttonSave, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap()
        );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

        .addGroup(layout.createSequentialGroup())
        .addContainerGap()
        .addComponent(panelAdd, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addContainerGap())
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(panelAdd, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(9, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

// quando é clicado em "Cancelar", fecha a janela interna "Adicionar livro"
private void buttonCancelActionPerformed(java.awt.event.ActionEvent evt) {
    closeWindow();
}

// quando uma tecla é solta no "textTitle", chama o método verifyText()
private void textNameKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// quando uma tecla é solta no "textAuthor", chama o método verifyText()
private void textUrlKeyReleased(java.awt.event.KeyEvent evt) {
    verifyText();
}

// adiciona um novo autor no banco de dados
private void buttonSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String name = textName.getText();
    String url = textUrl.getText();
    Integer publisherId = ((int)
ViewPublisher.tablePublisher.getValueAt(ViewPublisher.tablePublisher.getSelectedRow(), 0));

    Publisher publisher = new Publisher(publisherId, name, url);
    PublisherDAO.updatePublisher(publisher);

    Object[] options = { "Ok" };
    Icon figura = new ImageIcon
(getToolkit().createImage(getClass().getResource("../images/icon-done.png")));
    JOptionPane.showOptionDialog(null, "Editora atualizada!", "Editar editora",
JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE, figura, options, options[0]);
    closeWindow();
    ControllerView.readTablePublisher();
    ViewPublisher.buttonEdit.setEnabled(false);
    ViewPublisher.buttonDelete.setEnabled(false);
}

```

```

// limita a quantidade de caracteres em "Nome" para 30
private void textNameKeyTyped(java.awt.event.KeyEvent evt) {
    if ((textName.getText() + evt.getKeyChar()).length() > 30) {
        evt.consume();
    }
}

// limita a quantidade de caracteres em "URL" para 80 e proíbe espaço
private void textUrlKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (((textUrl.getText() + c).length() > 80) || c == KeyEvent.VK_SPACE) {
        evt.consume();
    }
}

// fecha a janela atual
private void closeWindow() {
    try {
        this.setClosed(true);
        ViewPublisher.editPublisherIsOpen = false;
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
}

// define a posição da janela interna no centro do programa
protected void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

/* verifica se existe texto nos campos "Título", "Autor", "Editora", "ISBN" e "Preço"
   caso todos tenham texto: habilita o botão "Adicionar"
   caso não: desabilita o botão "Adicionar" */
private void verifyText() {
    String textT = textName.getText();
    String textA = textUrl.getText();

    if (textT.isBlank() || textA.isBlank()) {
        buttonSave.setEnabled(false);
    } else {
        buttonSave.setEnabled(true);
    }
}

// Variables declaration - do not modify
private javax.swing.JButton buttonCancel;
private javax.swing.JButton buttonSave;
private javax.swing.JLabel labelId;
private javax.swing.JLabel labelName;
private javax.swing.JLabel labelUrl;
private javax.swing.JPanel panelAdd;

```

```

protected javax.swing.JTextField textId;
protected javax.swing.JTextField textName;
protected javax.swing.JTextField textUrl;
// End of variables declaration
}

```

## ViewGeneral.java

```

package view;

import controller.ControllerView;
import java.awt.Dimension;
import java.beans.PropertyVetoException;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;
import javax.swing.table.DefaultTableModel;

public class ViewGeneral extends javax.swing.JInternalFrame {

    protected ViewGeneral() {
        initComponents();
        buttonClean.setEnabled(false);
        buttonSearch.setEnabled(false);
        buttonGroupType.add(radioAuthor);
        buttonGroupType.add(radioTitle);
        buttonGroupType.add(radioPublisher);
        buttonGroupType.add(radiolsbn);
        radioAuthor.setFocusPainted(false);
        radioTitle.doClick();
        tableGeneral.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewLivrariaAmazonia.generallIsOpen = false;
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        buttonGroupType = new javax.swing.ButtonGroup();
        panelLivraria = new javax.swing.JPanel();
        radioTitle = new javax.swing.JRadioButton();
        radioAuthor = new javax.swing.JRadioButton();
        radioPublisher = new javax.swing.JRadioButton();
        radiolsbn = new javax.swing.JRadioButton();
        buttonSearch = new javax.swing.JButton();
        buttonClean = new javax.swing.JButton();
        buttonClose = new javax.swing.JButton();
        paneBooklist = new javax.swing.JScrollPane();
        tableGeneral = new javax.swing.JTable();
        textSearch = new javax.swing.JTextField();
    }

```

```

buttonShowAll = new javax.swing.JButton();

setClosable(true);
setIconifiable(true);
setMaximizable(true);
setResizable(true);
setTitle("Busca Geral");

panelLivraria.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0,
0, 0)));

radioTitle.setText("Título");

radioAuthor.setText("Autor");

radioPublisher.setText("Editora");

radiolsbn.setText("ISBN");

buttonSearch.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
search.png"))); // NOI18N
buttonSearch.setText("Buscar");
buttonSearch.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonSearchActionPerformed(evt);
    }
});

buttonClean.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
clear.png"))); // NOI18N
buttonClean.setText("Limpar");
buttonClean.setToolTipText("");
buttonClean.setMinimumSize(new java.awt.Dimension(134, 22));
buttonClean.setPreferredSize(new java.awt.Dimension(134, 22));
buttonClean.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonCleanActionPerformed(evt);
    }
});

buttonClose.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
exit.png"))); // NOI18N
buttonClose.setText("Fechar");
buttonClose.setToolTipText("");
buttonClose.setMinimumSize(new java.awt.Dimension(134, 22));
buttonClose.setPreferredSize(new java.awt.Dimension(134, 22));
buttonClose.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonCloseActionPerformed(evt);
    }
});

tableGeneral.setAutoCreateRowSorter(true);

```



```

tableGeneral.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

        },
        new String [] {
            "Livro", "ISBN", "Autor", "Preço", "Editora"
        }
    )
);
tableGeneral.getTableHeader().setReorderingAllowed(false);
paneBooklist.setViewportView(tableGeneral);

textSearch.setToolTipText("Digite o que deseja buscar");
textSearch.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyReleased(java.awt.event.KeyEvent evt) {
        textSearchKeyReleased(evt);
    }
});

buttonShowAll.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-show.png"))); // NOI18N
buttonShowAll.setText("Mostrar tudo");
buttonShowAll.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonShowAllActionPerformed(evt);
    }
});

javax.swing.GroupLayout panelLivrariaLayout = new
javax.swing.GroupLayout(panelLivraria);
panelLivraria.setLayout(panelLivrariaLayout);
panelLivrariaLayout.setHorizontalGroup(
    panelLivrariaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelLivrariaLayout.createSequentialGroup()
            .addContainerGap()
            .addGroup(panelLivrariaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(paneBooklist, javax.swing.GroupLayout.DEFAULT_SIZE, 685,
Short.MAX_VALUE)
                .addGroup(panelLivrariaLayout.createSequentialGroup()
                    .addComponent(radioTitle, javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(radioAuthor, javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(radioPublisher, javax.swing.GroupLayout.PREFERRED_SIZE,
69, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

        .addComponent(radiolsbn, javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 0, Short.MAX_VALUE))
        .addGroup(panelLivreriaLayout.createSequentialGroup())
        .addComponent(textSearch)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonSearch, javax.swing.GroupLayout.PREFERRED_SIZE,
120, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelLivreriaLayout.createSequentialGroup())
        .addGap(0, 0, Short.MAX_VALUE)
        .addComponent(buttonShowAll)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonClean, javax.swing.GroupLayout.PREFERRED_SIZE, 119,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(buttonClose, javax.swing.GroupLayout.PREFERRED_SIZE, 120,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap()
    );

```

```

    panelLivreriaLayout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {buttonClean, buttonClose, buttonShowAll});

```

```

    panelLivreriaLayout.setVerticalGroup(
        panelLivreriaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelLivreriaLayout.createSequentialGroup())
        .addContainerGap()
        .addGroup(panelLivreriaLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
            .addComponent(radioTitle, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(radioAuthor, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(radioPublisher, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(radiolsbn, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(panelLivreriaLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                .addComponent(textSearch, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(buttonSearch, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(paneBooklist, javax.swing.GroupLayout.DEFAULT_SIZE, 240,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(panelLivreriaLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                    .addComponent(buttonClean, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

        .addComponent(buttonClose, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonShowAll))
        .addContainerGap()
    );

    panelLivrariaLayout.linkSize(javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] {buttonClean, buttonClose, buttonSearch, buttonShowAll, textSearch});

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(panelLivraria, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap()
        );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(panelLivraria, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap()
        );
    pack();
} // </editor-fold>

/* quando uma tecla é solta dentro do campo de busca
verifica se existe texto no campo de busca
caso sim: habilita o botão "Pesquisar"
caso não: desabilita o botão "Pesquisar" */
private void textSearchKeyReleased(java.awt.event.KeyEvent evt) {
    String text = textSearch.getText();

    if (text.isBlank()) {
        buttonSearch.setEnabled(false);
    } else {
        buttonSearch.setEnabled(true);
    }
}

// quando é clicado em "Fechar", fecha a janela interna "Buscar livro"
private void buttonCloseActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        this.setClosed(true);
        ViewLivrariaAmazonia.generallsOpen = false;
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
}

```

```

}

// realiza a pesquisa de acordo com o tipo selecionado
private void buttonSearchActionPerformed(java.awt.event.ActionEvent evt) {
    String search = textSearch.getText();

    if (radioTitle.isSelected()) {
        ControllerView.readTableTitle(search);
    } else if (radioAuthor.isSelected()) {
        ControllerView.readTableAuthor(search);
    } else if (radioPublisher.isSelected()) {
        ControllerView.readTablePublisher(search);
    } else if (radioIsbn.isSelected()) {
        ControllerView.readTableIsbn(search);
    }

    buttonClean.setEnabled(true);
    buttonShowAll.setEnabled(true);
}

// limpa a busca
private void buttonCleanActionPerformed(java.awt.event.ActionEvent evt) {
    buttonClean.setEnabled(false);
    buttonSearch.setEnabled(false);
    buttonShowAll.setEnabled(true);
    DefaultTableModel modelo = (DefaultTableModel) tableGeneral.getModel();
    modelo.setNumRows(0);
    textSearch.setText("");
}

private void buttonShowAllActionPerformed(java.awt.event.ActionEvent evt) {
    ControllerView.readTableGeneral();
    buttonClean.setEnabled(true);
    buttonShowAll.setEnabled(false);
}

// define a posição da janela interna no centro do programa
protected void setPositionCenter() {
    Dimension d = this.getDesktopPane().getSize();
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

// Variables declaration - do not modify
public static javax.swing.JButton buttonClean;
private javax.swing.JButton buttonClose;
private javax.swing.ButtonGroup buttonGroupType;
private javax.swing.JButton buttonSearch;
private javax.swing.JButton buttonShowAll;
private javax.swing.JScrollPane paneBooklist;
private javax.swing.JPanel panelLivraria;
private javax.swing.JRadioButton radioAuthor;
private javax.swing.JRadioButton radioIsbn;
private javax.swing.JRadioButton radioPublisher;

```

```

private javax.swing.JRadioButton radioTitle;
public static javax.swing.JTable tableGeneral;
private javax.swing.JTextField textSearch;
// End of variables declaration
}

ViewLivrariaAmazonia.java

```

```

package view;

import java.awt.Graphics;
import java.awt.Image;
import javax.swing.ImageIcon;

public class ViewLivrariaAmazonia extends javax.swing.JFrame {

    static boolean aboutIsOpen = false;
    static boolean bookIsOpen = false;
    static boolean authorIsOpen = false;
    static boolean generalIsOpen = false;
    static boolean publisherIsOpen = false;

    public ViewLivrariaAmazonia() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        ImageIcon icon = new ImageIcon(getClass().getResource("/images/background.jpg"));
        Image img = icon.getImage();
        desktopAmazonia = new javax.swing.JDesktopPane() {

            public void paintComponent(Graphics g) {
                g.drawImage(img,0,0,2560,1440,this);
            }
        };
        labelWelcome = new javax.swing.JLabel();
        panelButtons = new javax.swing.JPanel();
        buttonBooks = new javax.swing.JButton();
        buttonAuthors = new javax.swing.JButton();
        buttonPublishers = new javax.swing.JButton();
        buttonGeneral = new javax.swing.JButton();
        menuBarra = new javax.swing.JMenuBar();
        menuLivraria = new javax.swing.JMenu();
        menuLivrariaGeneral = new javax.swing.JMenuItem();
        menuLivrariaBook = new javax.swing.JMenuItem();
        menuLivrariaAuthor = new javax.swing.JMenuItem();
        menuLivrariaPublisher = new javax.swing.JMenuItem();
        menuAjuda = new javax.swing.JMenu();
        menuAjudaSobre = new javax.swing.JMenuItem();
    }
}

```

```

menuAjudaSair = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Livraria Amazônia");
setBounds(new java.awt.Rectangle(0, 0, 0, 0));
setLocation(new java.awt.Point(0, 0));
setLocationByPlatform(true);
setMinimumSize(new java.awt.Dimension(600, 500));
setName("Livraria Amazônia"); // NOI18N
setSize(new java.awt.Dimension(1000, 500));

labelWelcome.setFont(new java.awt.Font("Rockwell Condensed", 1, 36)); // NOI18N
labelWelcome.setForeground(new java.awt.Color(0, 102, 51));
labelWelcome.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
labelWelcome.setText("Bem vindo(a) à Livraria Amazônia!");

panelButtons.setBackground(new java.awt.Color(0, 102, 0));

buttonBooks.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
buttonBooks.setText("Livros");
buttonBooks.setPreferredSize(new java.awt.Dimension(180, 38));
buttonBooks.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonBooksActionPerformed(evt);
    }
});

buttonAuthors.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
buttonAuthors.setText("Autores");
buttonAuthors.setPreferredSize(new java.awt.Dimension(180, 38));
buttonAuthors.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonAuthorsActionPerformed(evt);
    }
});

buttonPublishers.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
buttonPublishers.setText("Editoras");
buttonPublishers.setPreferredSize(new java.awt.Dimension(180, 38));
buttonPublishers.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonPublishersActionPerformed(evt);
    }
});

buttonGeneral.setFont(new java.awt.Font("Segoe UI", 0, 24)); // NOI18N
buttonGeneral.setText("Buscar");
buttonGeneral.setPreferredSize(new java.awt.Dimension(180, 38));
buttonGeneral.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonGeneralActionPerformed(evt);
    }
});

```

```

        javax.swing.GroupLayout panelButtonsLayout = new
javax.swing.GroupLayout(panelButtons);
        panelButtons.setLayout(panelButtonsLayout);
        panelButtonsLayout.setHorizontalGroup(
            panelButtonsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelButtonsLayout.createSequentialGroup()
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGroup(panelButtonsLayout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.TRAILING)
                        .addGroup(panelButtonsLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING, false)
                            .addComponent(buttonPublishers, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(buttonGeneral, javax.swing.GroupLayout.PREFERRED_SIZE,
210, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(panelButtonsLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING, false)
                            .addComponent(buttonAuthors, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(buttonBooks, javax.swing.GroupLayout.PREFERRED_SIZE, 210,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addContainerGap())
                );
        panelButtonsLayout.setVerticalGroup(
            panelButtonsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelButtonsLayout.createSequentialGroup()
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(buttonBooks, javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(buttonAuthors, javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(buttonPublishers, javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(buttonGeneral, javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap())
                );

        desktopAmazonia.setLayer(labelWelcome, javax.swing.JLayeredPane.DEFAULT_LAYER);
        desktopAmazonia.setLayer(panelButtons, javax.swing.JLayeredPane.DEFAULT_LAYER);

        javax.swing.GroupLayout desktopAmazoniaLayout = new
javax.swing.GroupLayout(desktopAmazonia);
        desktopAmazonia.setLayout(desktopAmazoniaLayout);
        desktopAmazoniaLayout.setHorizontalGroup(

desktopAmazoniaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

        .addGroup(desktopAmazoniaLayout.createSequentialGroup())
        .addContainerGap()
        .addComponent(labelWelcome, javax.swing.GroupLayout.DEFAULT_SIZE, 746,
Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
desktopAmazoniaLayout.createSequentialGroup())
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(panelButtons, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    desktopAmazoniaLayout.setVerticalGroup(

desktopAmazoniaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
desktopAmazoniaLayout.createSequentialGroup())
        .addGap(18, 18, 18)
        .addComponent(labelWelcome, javax.swing.GroupLayout.PREFERRED_SIZE, 87,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(panelButtons, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(92, Short.MAX_VALUE))
    );

    menuLivraria.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
open-book.png"))); // NOI18N
    menuLivraria.setText("Livraria");
    menuLivraria.setFocusable(false);
    menuLivraria.setRequestFocusEnabled(false);

    menuLivrariaGeneral.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-search.png"))); // NOI18N
    menuLivrariaGeneral.setText("Buscar");
    menuLivrariaGeneral.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            menuLivrariaGeneralActionPerformed(evt);
        }
    });
    menuLivraria.add(menuLivrariaGeneral);

    menuLivrariaBook.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-open-book.png"))); // NOI18N
    menuLivrariaBook.setText("Livro");
    menuLivrariaBook.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            menuLivrariaBookActionPerformed(evt);
        }
    });
    menuLivraria.add(menuLivrariaBook);

    menuLivrariaAuthor.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-writer.png"))); // NOI18N

```



```

menuLivrariaAuthor.setText("Autor");
menuLivrariaAuthor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        menuLivrariaAuthorActionPerformed(evt);
    }
});
menuLivraria.add(menuLivrariaAuthor);

menuLivrariaPublisher.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-publisher.png"))); // NOI18N
menuLivrariaPublisher.setText("Editora");
menuLivrariaPublisher.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        menuLivrariaPublisherActionPerformed(evt);
    }
});
menuLivraria.add(menuLivrariaPublisher);

menuBarra.add(menuLivraria);

menuAjuda.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
about.png"))); // NOI18N
menuAjuda.setText("Ajuda");

menuAjudaSobre.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/icon-about.png"))); // NOI18N
menuAjudaSobre.setText("Sobre");
menuAjudaSobre.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        menuAjudaSobreActionPerformed(evt);
    }
});
menuAjuda.add(menuAjudaSobre);

menuAjudaSair.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
exit.png"))); // NOI18N
menuAjudaSair.setText("Sair");
menuAjudaSair.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        menuAjudaSairActionPerformed(evt);
    }
});
menuAjuda.add(menuAjudaSair);

menuBarra.add(menuAjuda);

setJMenuBar(menuBarra);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(desktopAmazonia)

```

```

);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(desktopAmazonia)
);

pack();
setLocationRelativeTo(null);
} // </editor-fold>

// fecha o programa pelo "Sair" no menu superior
private void menuAjudaSairActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

// abre a janela interna "Sobre" pelo menu superior, com limite de 1 janela aberta por vez
private void menuAjudaSobreActionPerformed(java.awt.event.ActionEvent evt) {
    if (!aboutIsOpen) {
        ViewAbout viewAbout = new ViewAbout();
        desktopAmazonia.add(viewAbout);
        viewAbout.setVisible(true);
        viewAbout.setPositionCenter();
        aboutIsOpen = true;
    }
}

// abre a janela interna "Livros", com limite de 1 janela aberta por vez
private void menuLivrariaBookActionPerformed(java.awt.event.ActionEvent evt) {
    if (!bookIsOpen) {
        ViewBook viewBook = new ViewBook();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewBook);
        viewBook.setVisible(true);
        viewBook.setPositionCenter();
        bookIsOpen = true;
    }
}

// abre a janela interna "Autores", com limite de 1 janela aberta por vez
private void menuLivrariaAuthorActionPerformed(java.awt.event.ActionEvent evt) {
    if (!authorIsOpen) {
        ViewAuthor viewAuthor = new ViewAuthor();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewAuthor);
        viewAuthor.setVisible(true);
        viewAuthor.setPositionCenter();
        authorIsOpen = true;
    }
}

// abre a janela interna "Editoras", com limite de 1 janela aberta por vez
private void menuLivrariaPublisherActionPerformed(java.awt.event.ActionEvent evt) {
    if (!publisherIsOpen) {
        ViewPublisher viewPublisher = new ViewPublisher();
    }
}

```

```

        ViewLivrariaAmazonia.desktopAmazonia.add(viewPublisher);
        viewPublisher.setVisible(true);
        viewPublisher.setPositionCenter();
        publisherIsOpen = true;
    }
}

// abre a janela interna "Buscar" , com limite de 1 janela aberta por vez
private void menuLivrariaGeneralActionPerformed(java.awt.event.ActionEvent evt) {
    if (!generalIsOpen) {
        ViewGeneral viewSearch = new ViewGeneral();
        desktopAmazonia.add(viewSearch);
        viewSearch.setVisible(true);
        viewSearch.setPositionCenter();
        generalIsOpen = true;
    }
}

// abre a janela interna "Livros" pelo botão, com limite de 1 janela aberta por vez
private void buttonBooksActionPerformed(java.awt.event.ActionEvent evt) {
    if (!bookIsOpen) {
        ViewBook viewBook = new ViewBook();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewBook);
        viewBook.setVisible(true);
        viewBook.setPositionCenter();
        bookIsOpen = true;
    }
}

// abre a janela interna "Autores" pelo botão, com limite de 1 janela aberta por vez
private void buttonAuthorsActionPerformed(java.awt.event.ActionEvent evt) {
    if (!authorIsOpen) {
        ViewAuthor viewAuthor = new ViewAuthor();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewAuthor);
        viewAuthor.setVisible(true);
        viewAuthor.setPositionCenter();
        authorIsOpen = true;
    }
}

// abre a janela interna "Editoras" pelo botão, com limite de 1 janela aberta por vez
private void buttonPublishersActionPerformed(java.awt.event.ActionEvent evt) {
    if (!publisherIsOpen) {
        ViewPublisher viewPublisher = new ViewPublisher();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewPublisher);
        viewPublisher.setVisible(true);
        viewPublisher.setPositionCenter();
        publisherIsOpen = true;
    }
}

// abre a janela interna "geral" pelo botão, com limite de 1 janela aberta por vez
private void buttonGeneralActionPerformed(java.awt.event.ActionEvent evt) {

```

```

        if (!generallsOpen) {
            ViewGeneral viewSearch = new ViewGeneral();
            desktopAmazonia.add(viewSearch);
            viewSearch.setVisible(true);
            viewSearch.setPositionCenter();
        }
    }

    // MAIN
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
                javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

            java.util.logging.Logger.getLogger(ViewLivrariaAmazonia.class.getName()).log(java.util.logging.L
                evel.SEVERE, null, ex);
        } catch (InstantiationException ex) {

            java.util.logging.Logger.getLogger(ViewLivrariaAmazonia.class.getName()).log(java.util.logging.L
                evel.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

            java.util.logging.Logger.getLogger(ViewLivrariaAmazonia.class.getName()).log(java.util.logging.L
                evel.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

            java.util.logging.Logger.getLogger(ViewLivrariaAmazonia.class.getName()).log(java.util.logging.L
                evel.SEVERE, null, ex);
        }
    }

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            ViewLivrariaAmazonia vla = new ViewLivrariaAmazonia();
            vla.setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton buttonAuthors;
private javax.swing.JButton buttonBooks;
private javax.swing.JButton buttonGeneral;
private javax.swing.JButton buttonPublishers;
protected static javax.swing.JDesktopPane desktopAmazonia;
private javax.swing.JLabel labelWelcome;

```

```

private javax.swing.JMenu menuAjuda;
private javax.swing.JMenuItem menuAjudaSair;
private javax.swing.JMenuItem menuAjudaSobre;
private javax.swing.JMenuBar menuBarra;
private javax.swing.JMenu menuLivraria;
private javax.swing.JMenuItem menuLivrariaAuthor;
private javax.swing.JMenuItem menuLivrariaBook;
private javax.swing.JMenuItem menuLivrariaGeneral;
private javax.swing.JMenuItem menuLivrariaPublisher;
private javax.swing.JPanel panelButtons;
// End of variables declaration
}

```

#### ViewPublisher.java

```

package view;

import controller.ControllerView;
import java.awt.Dimension;
import java.beans.PropertyVetoException;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.event.InternalFrameAdapter;
import javax.swing.event.InternalFrameEvent;
import javax.swing.table.DefaultTableModel;
import model.bean.Publisher;
import model.dao.PublisherDAO;

public class ViewPublisher extends javax.swing.JInternalFrame {

    static boolean addPublisherIsOpen = false;
    static boolean editPublisherIsOpen = false;

    protected ViewPublisher() {
        initComponents();
        buttonEdit.setEnabled(false);
        buttonDelete.setEnabled(false);
        buttonShowAll.setEnabled(false);
        buttonSearch.setEnabled(false);
        ControllerView.readTablePublisher();
        tablePublisher.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        addInternalFrameListener(new InternalFrameAdapter(){
            public void internalFrameClosing(InternalFrameEvent e) {
                ViewLivrariaAmazonia.publisherIsOpen = false;
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

```

```

panelEdit = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
tablePublisher = new javax.swing.JTable();
buttonAdd = new javax.swing.JButton();
buttonEdit = new javax.swing.JButton();
buttonDelete = new javax.swing.JButton();
buttonClose = new javax.swing.JButton();
textSearch = new javax.swing.JTextField();
buttonSearch = new javax.swing.JButton();
buttonClear = new javax.swing.JButton();
buttonShowAll = new javax.swing.JButton();

setClosable(true);
setIconifiable(true);
setMaximizable(true);
setResizable(true);
setTitle("Editoras");
setPreferredSize(new java.awt.Dimension(670, 400));

panelEdit.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0,
0)));

tablePublisher.setAutoCreateRowSorter(true);
tablePublisher.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

        },
        new String [] {
            "ID", "Nome", "URL"
        }
    )
    {
        public boolean isCellEditable(int row, int column) {
            return false;
        }
    }
);
tablePublisher.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        tablePublisherMouseClicked(evt);
    }
});
jScrollPane1.setViewportView(tablePublisher);

buttonAdd.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-add-
book.png"))); // NOI18N
buttonAdd.setText("Adicionar");
buttonAdd.setMinimumSize(new java.awt.Dimension(134, 22));
buttonAdd.setPreferredSize(new java.awt.Dimension(134, 22));
buttonAdd.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        buttonAddActionPerformed(evt);
    }
}

```

```

    });

    buttonEdit.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
pencil.png"))); // NOI18N
    buttonEdit.setText("Editar");
    buttonEdit.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonEdit.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonEdit.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonEditActionPerformed(evt);
        }
    });

    buttonDelete.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
delete.png"))); // NOI18N
    buttonDelete.setText("Excluir");
    buttonDelete.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonDelete.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonDelete.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonDeleteActionPerformed(evt);
        }
    });

    buttonClose.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
exit.png"))); // NOI18N
    buttonClose.setText("Fechar");
    buttonClose.setMinimumSize(new java.awt.Dimension(134, 22));
    buttonClose.setPreferredSize(new java.awt.Dimension(134, 22));
    buttonClose.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonCloseActionPerformed(evt);
        }
    });

    textSearch.addKeyListener(new java.awt.event.KeyAdapter() {
        public void keyReleased(java.awt.event.KeyEvent evt) {
            textSearchKeyReleased(evt);
        }
    });

    buttonSearch.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
search.png"))); // NOI18N
    buttonSearch.setText("Buscar");
    buttonSearch.setPreferredSize(new java.awt.Dimension(119, 22));
    buttonSearch.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            buttonSearchActionPerformed(evt);
        }
    });

    buttonClear.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/icon-
clear.png"))); // NOI18N

```





```

        panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(panelEditLayout.createSequentialGroup()
        .addContainerGap()
        .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(buttonSearch, javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(textSearch, javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonClear, javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonShowAll)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 265,
        Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(panelEditLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(buttonAdd, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
        javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonClose, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
        javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonDelete, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
        javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(buttonEdit, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
        javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap())
    );

```

```

    panelEditLayout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
    {buttonAdd, buttonClear, buttonClose, buttonDelete, buttonEdit, buttonSearch, buttonShowAll,
    textSearch});

```

```

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(panelEdit, javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addContainerGap())
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(panelEdit, javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGap(7, 7, 7))
    );

```

```

pack();
} // </editor-fold>

// quando um item da tabela é selecionado, ativa os botões "buttonEdit" e "buttonDelete"
private void tablePublisherMouseClicked(java.awt.event.MouseEvent evt) {
    buttonEdit.setEnabled(true);
    buttonDelete.setEnabled(true);
}

// quando é clicado em "Cancelar", fecha a janela interna "Editoras"
private void buttonCloseActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        this.setClosed(true);
        ViewLivrariaAmazonia.publisherIsOpen = false;
    } catch (PropertyVetoException ex) {
        System.err.println("Closing Exception");
    }
}

// quando é clicado em "Adicionar" chama a view "ViewAddPublisher", limitando para 1 janela
dessa aberta
private void buttonAddActionPerformed(java.awt.event.ActionEvent evt) {
    if (!addPublisherIsOpen) {
        ViewAddPublisher viewAddPublisher = new ViewAddPublisher();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewAddPublisher);
        viewAddPublisher.setVisible(true);
        viewAddPublisher.setPositionCenter();
        addPublisherIsOpen = true;
    }
}

// quando é clicado em "Excluir" exclui a editora do banco de dados e atualiza a tabela
private void buttonDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    Object[] options = { "Sim", "Não" };
    Icon figura = new ImageIcon(
(getToolkit().createImage(getClass().getResource("../images/icon-warning.png"))));
    int option = JOptionPane.showOptionDialog(null, "Caso exclua essa editora, todos os livros
vinculados a ela também serão excluídos.\nGostaria de continuar mesmo assim?", "Excluir
editora", JOptionPane.DEFAULT_OPTION, JOptionPane.WARNING_MESSAGE, figura, options,
options[1]);

    if (option == 0) {
        Publisher publisher = new Publisher();
        PublisherDAO publisherDAO = new PublisherDAO();

        publisher.setId((int) tablePublisher.getValueAt(tablePublisher.getSelectedRow(), 0));
        publisherDAO.deletePublisher(publisher);
        ControllerView.readTablePublisher();
        buttonDelete.setEnabled(false);
        buttonEdit.setEnabled(false);
    }
}

```

```

/* quando é clicado em "editar" chama a internalFrame "viewPublisher" já com as
informações da editora selecionada, limitando para 1 janela dessa aberta */
private void buttonEditActionPerformed(java.awt.event.ActionEvent evt) {
    if (!editPublisherIsOpen) {
        ViewEditPublisher viewEditPublisher = new ViewEditPublisher();
        ViewLivrariaAmazonia.desktopAmazonia.add(viewEditPublisher);
        viewEditPublisher.setVisible(true);
        viewEditPublisher.setPositionCenter();
        viewEditPublisher.textId.setEnabled(false);

        String name = ((String) tablePublisher.getValueAt(tablePublisher.getSelectedRow(), 1));
        viewEditPublisher.textName.setText(name);

        String url = ((String) tablePublisher.getValueAt(tablePublisher.getSelectedRow(), 2));
        viewEditPublisher.textUrl.setText(url);

        Integer id = ((int) tablePublisher.getValueAt(tablePublisher.getSelectedRow(), 0));
        viewEditPublisher.textId.setText(id.toString());
        editPublisherIsOpen = true;
    }
}

private void textSearchKeyReleased(java.awt.event.KeyEvent evt) {
    String text = textSearch.getText();

    if (text.isBlank()) {
        buttonSearch.setEnabled(false);
    } else {
        buttonSearch.setEnabled(true);
    }
}

// realiza a pesquisa de editora
private void buttonSearchActionPerformed(java.awt.event.ActionEvent evt) {
    String search = textSearch.getText();
    ControllerView.readTablePublisherGeneral(search);
    buttonClear.setEnabled(true);
    buttonShowAll.setEnabled(true);
}

private void buttonClearActionPerformed(java.awt.event.ActionEvent evt) {
    buttonClear.setEnabled(false);
    buttonSearch.setEnabled(false);
    buttonShowAll.setEnabled(true);
    DefaultTableModel modelo = (DefaultTableModel) tablePublisher.getModel();
    modelo.setNumRows(0);
    textSearch.setText("");
}

private void buttonShowAllActionPerformed(java.awt.event.ActionEvent evt) {
    ControllerView.readTablePublisher();
    buttonClear.setEnabled(true);
}

```

```

        buttonShowAll.setEnabled(false);
    }

    // define a posição da janela interna no centro do programa
    protected void setPositionCenter() {
        Dimension d = this.getDesktopPane().getSize();
        this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
    }

    // Variables declaration - do not modify
    private javax.swing.JButton buttonAdd;
    private javax.swing.JButton buttonClear;
    private javax.swing.JButton buttonClose;
    protected static javax.swing.JButton buttonDelete;
    protected static javax.swing.JButton buttonEdit;
    private javax.swing.JButton buttonSearch;
    private javax.swing.JButton buttonShowAll;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JPanel panelEdit;
    public static javax.swing.JTable tablePublisher;
    private javax.swing.JTextField textSearch;
    // End of variables declaration
}

```

## **Ficha de Atividades Práticas Supervisionadas**

Ordem das fichas de horários:

1. Carlos Eduardo dos Santos Ferreira
2. Gabriel Menezes de Antonio
3. Gustavo Henrique dos Santos Faria
4. Mayara Marques Pereira de Souza

## FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Carlos Eduardo dos Santos Ferreira TURMA: CC4P12 RA: N6401C-7

CURSO: Ciência da Computação CAMPUS: Campinas SEMESTRE: 4º TURNO: Noturno

CÓDIGO DA ATIVIDADE: 77B1 SEMESTRE: 4º Semestre ANO GRADE: 2º Ano

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
03/10/2021	Reunião com grupo	02:00:00	Carlos Eduardo		
03/11/2021	Pesquisa para o desenvolvimento do relatório	05:15:00	Carlos Eduardo		
05/11/2021	Pesquisa para o desenvolvimento do relatório	04:45:00	Carlos Eduardo		
06/11/2021	Reunião para discutir como as telas estão sendo implementadas	04:00:00	Carlos Eduardo		
06/11/2021	Desenvolvimento da introdução do relatório	06:30:00	Carlos Eduardo		
07/11/2021	Desenvolvimento da introdução do relatório	04:45:00	Carlos Eduardo		
07/11/2021	Desenvolvimento do referencial teórico do relatório	08:55:00	Carlos Eduardo		
07/11/2021	Reunião para discutirmos o design das telas	03:00:00	Carlos Eduardo		
09/11/2021	Desenvolvimento da parte de desenvolvimento do relatório	06:35:00	Carlos Eduardo		
11/11/2021	Desenvolvimento da parte de desenvolvimento do relatório	05:45:00	Carlos Eduardo		
12/11/2021	Desenvolvimento da parte de resultados do relatório	06:18:00	Carlos Eduardo		
13/11/2021	Desenvolvimento da parte de resultados do relatório	03:00:00	Carlos Eduardo		
15/11/2021	Desenvolvimento da parte de considerações finais do relatório	04:22:00	Carlos Eduardo		
17/11/2021	Reunião para discutirmos os resultados do relatório	02:30:00	Carlos Eduardo		
17/11/2021	Revisão do relatório	05:15:00	Carlos Eduardo		
18/11/2021	Revisão do relatório	03:30:00	Carlos Eduardo		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 76:25:00

AVALIAÇÃO: \_\_\_\_\_

Aprovado ou Reprovado

NOTA: \_\_\_\_\_

DATA: \_\_\_\_/\_\_\_\_/\_\_\_\_

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

## FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Gabriel Menezes de Antonio

TURMA: CC4Q12

RA: F13GJI-6

CURSO: Ciência da Computação

CAMPUS: Campinas


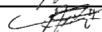
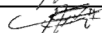
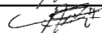

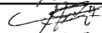
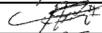

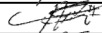
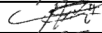


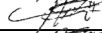
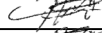
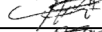

SEMESTRE: 4º

TURNO: Noturno

CÓDIGO DA ATIVIDADE: 77B1

SEMESTRE: 4º

ANO GRADE: 2º

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
03/10/2021	Reunião com o grupo	02:00:00			
10/10/2021	Desenvolvendo base do programa	04:30:00			
10/10/2021	Pesquisa de métodos de otimização de conexão com o BD	03:47:00			
17/10/2021	Desenvolvendo DAO	05:47:00			
25/10/2021	Estudo e aprimoramento de métodos de acesso ao BD com Java	07:34:00			
04/11/2021	Analisando e revisando código fonte	03:51:00			
04/11/2021	Auxiliando integrantes do grupo acessarem informações do BD	04:31:00			
05/11/2021	Criando classe de busca de dados no BD	05:49:00			
06/11/2021	Reunião para discutir como as telas estão sendo implementadas	04:00:00			
07/11/2021	Reunião para discutirmos o design das telas	03:00:00			
08/11/2021	Criação de métodos para busca e listagem de editoras e autores	04:27:00			
09/11/2021	Estudo e aprimoramento de conexões ao BD	06:15:00			
10/11/2021	Auxílio no desenvolvimento de tratamento de erros do programa	05:10:00			
11/11/2021	Teste do programa em busca de erros programáticos	06:43:00			
12/12/2021	Teste do programa em busca de erros de acesso ao BD	04:13:00			
13/12/2021	Teste final do programa em busca de erros e revisão do relatório	05:21:00			

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:

76:58:00

AVALIAÇÃO: \_\_\_\_\_

Aprovado ou Reprovado

NOTA: \_\_\_\_\_

DATA: \_\_\_\_/\_\_\_\_/\_\_\_\_

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



## FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: \_\_\_\_\_ Gustavo Henrique Dos Santos Faria \_\_\_\_\_ TURMA: \_\_\_\_\_ CC4Q12 \_\_\_\_\_ RA: \_\_\_\_\_ F22IFG-2 \_\_\_\_\_

CURSO: \_\_\_\_\_ Ciência Da Computação \_\_\_\_\_ CAMPUS: \_\_\_\_\_ UNIP SWIFT \_\_\_\_\_ SEMESTRE: \_\_\_\_\_ 4° Semestre \_\_\_\_\_ TURNO: \_\_\_\_\_ Noturno \_\_\_\_\_

CÓDIGO DA ATIVIDADE: \_\_\_\_\_ 77B1 \_\_\_\_\_ SEMESTRE: \_\_\_\_\_ 4° Semestre \_\_\_\_\_ ANO GRADE: \_\_\_\_\_ 2° Ano \_\_\_\_\_

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
03/10/2021	Reunião com grupo	4	<i>Gustavo</i>		
10/10/2021	Desenvolvendo base do programa	8	<i>Gustavo</i>		
22/10/2021	Desenvolvendo classes DAO	3	<i>Gustavo</i>		
25/10/2021	Arrumando problema com o driver JDBC	2	<i>Gustavo</i>		
26/10/2021	Modificando e testando DAO	4	<i>Gustavo</i>		
20/10/2021	Modificando e testando DAO	6	<i>Gustavo</i>		
02/11/2021	Modificando views	6	<i>Gustavo</i>		
03/11/2021	Adicionando ícones no projeto	5	<i>Gustavo</i>		
04/11/2021	Resolvendo bugs	5	<i>Gustavo</i>		
06/11/2021	Reunião para discutir como as telas estão sendo implementadas	4	<i>Gustavo</i>		
06/11/2021	Criando select e editando view	5	<i>Gustavo</i>		
07/11/2021	Reunião para discutirmos o design das telas	3	<i>Gustavo</i>		
09/11/2021	Adicionando comentário aos métodos	5	<i>Gustavo</i>		
10/11/2021	Alterando queries no sistema	5	<i>Gustavo</i>		
11/11/2021	Criando queries para views	5	<i>Gustavo</i>		
12/11/2021	Resolvendo bugs	5	<i>Gustavo</i>		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: \_\_\_\_\_ 75 Horas \_\_\_\_\_

AVALIAÇÃO: \_\_\_\_\_

Aprovado ou Reprovado

NOTA: \_\_\_\_\_

DATA: \_\_\_\_/\_\_\_\_/\_\_\_\_

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



**FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS**

NOME: MAYARA MARQUES PEREIRA DE SOUZA TURMA: CC4Q12 RA: N542DD-1  
 CURSO: CIÊNCIA DA COMPUTAÇÃO CAMPUS: SWIFT SEMESTRE: 4º TURNO: NOTURNO  
 CÓDIGO DA ATIVIDADE: 77B1 SEMESTRE: 4º ANO GRADE: 2º ANO

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
03/10	REUNIÃO DE PLANEJAMENTO COM O GRUPO	2h	May MPS		
06/10	INICIANDO CRIAÇÃO DA VIEW	6h 30m	May MPS		
10/10	DESENVOLVENDO BASE DO PROGRAMA	6h 30m	May MPS		
20/10	MELHORANDO A VIEW BASE	6h 30m	May MPS		
26/10	TROCANDO JANELAS POR JINTERNALEPANE	5h	May MPS		
04/11	CRIANDO NOVAS FRAMES DE ADD/EDIT/REMOVE	6h	May MPS		
06/11	ENTENDENDO E APLICANDO VTABLES	6h	May MPS		
06/11	REUNIÃO PARA DISCUTIR USO DE TELAS	4h	May MPS		
08/11	CRIANDO TELAS DE EDIÇÃO	7h	May MPS		
09/11	FINALIZANDO TELA DE LIVROS	6h	May MPS		
10/11	LIGANDO BUSCA COM BANCO DE DADOS	7h	May MPS		
12/11	ADICIONANDO BUSCAS INDIVIDUAIS E TESTANDO	7h	May MPS		
13/11	CORRIGINDO BUGS FINAIS	5h 30m	May MPS		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 75h

AVALIAÇÃO: \_\_\_\_\_

Aprovado ou Reprovado

NOTA: \_\_\_\_\_

DATA: \_\_\_\_/\_\_\_\_/\_\_\_\_

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO