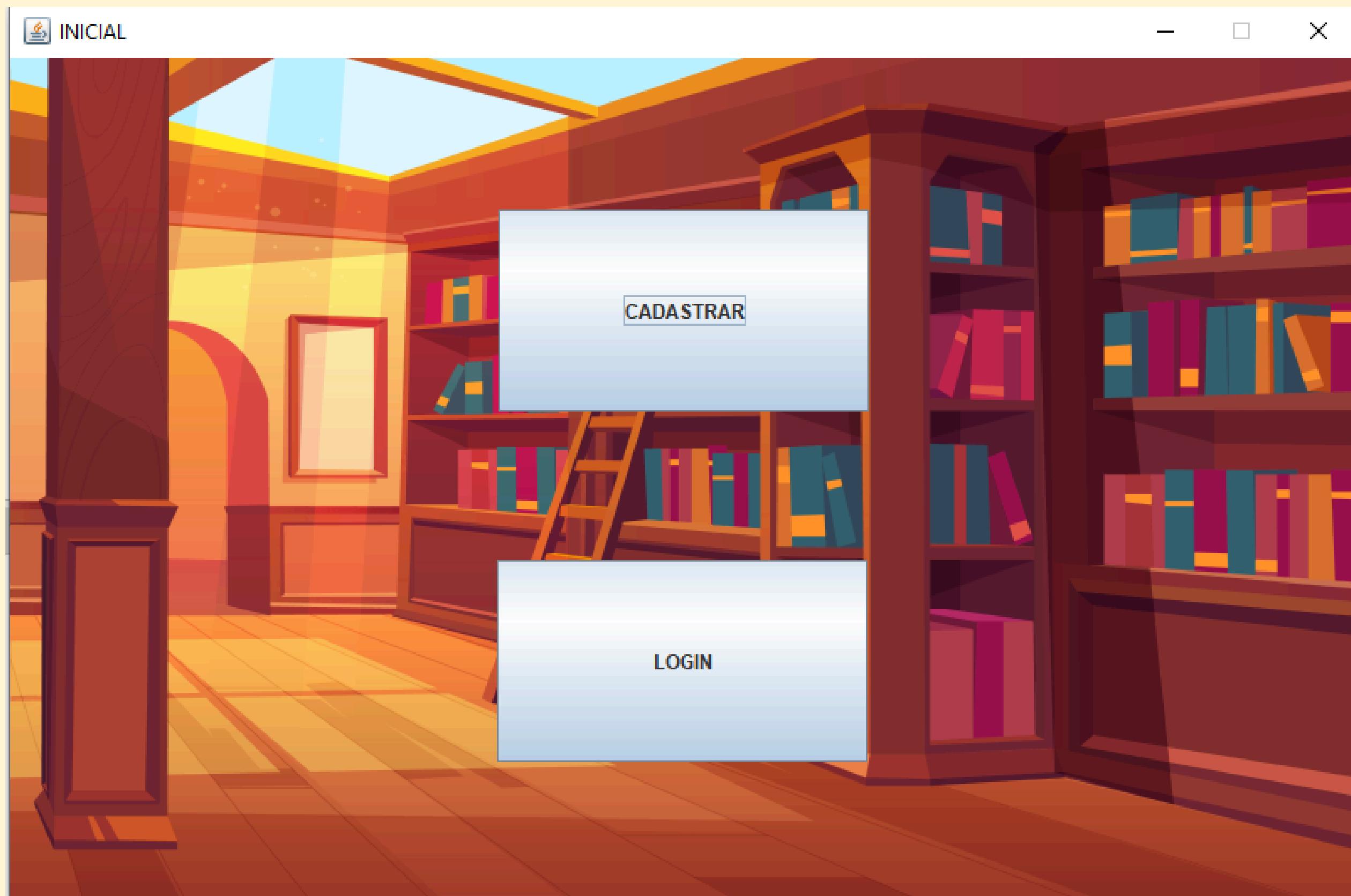


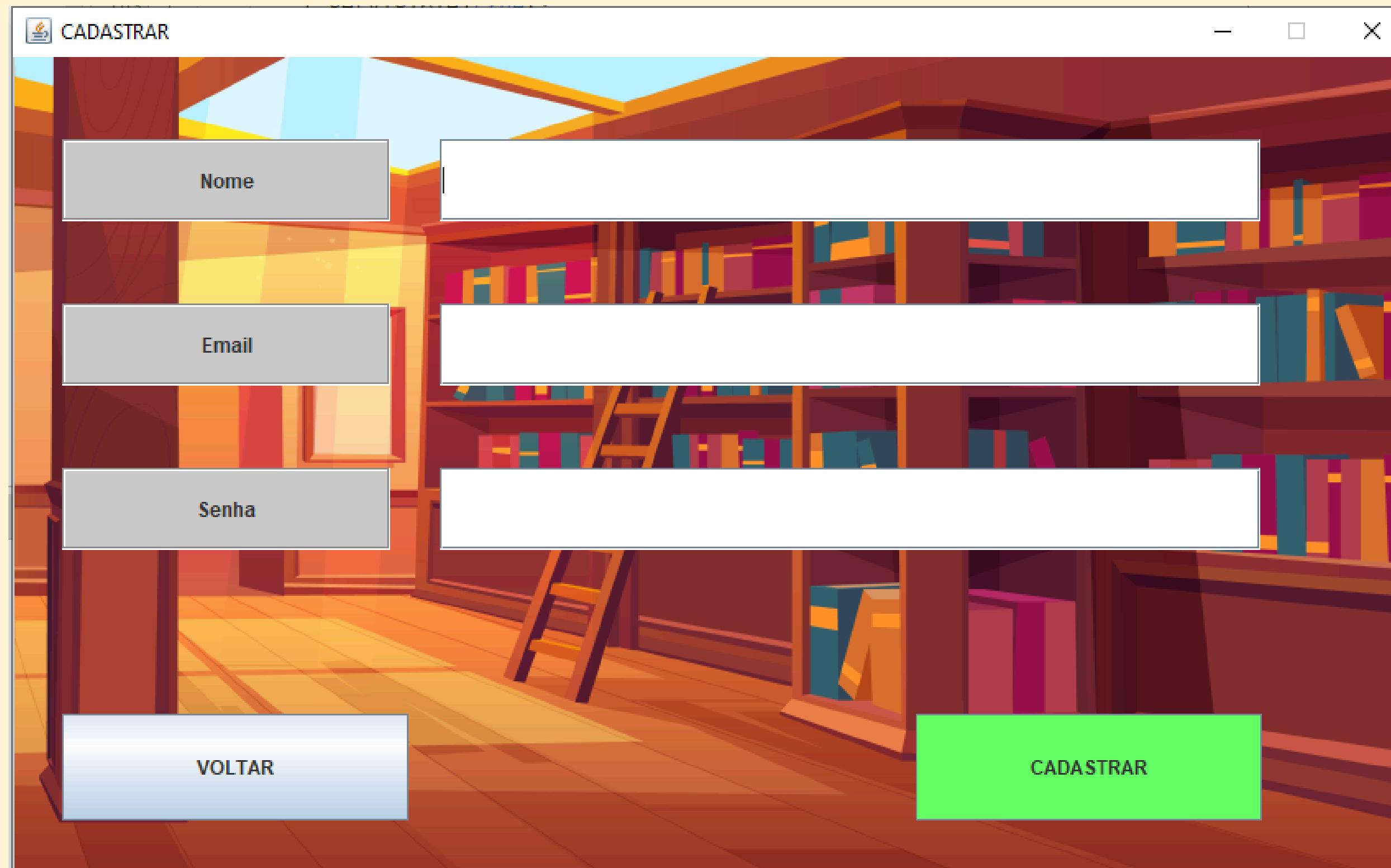
# SISTEMA DE BIBLIOTECA



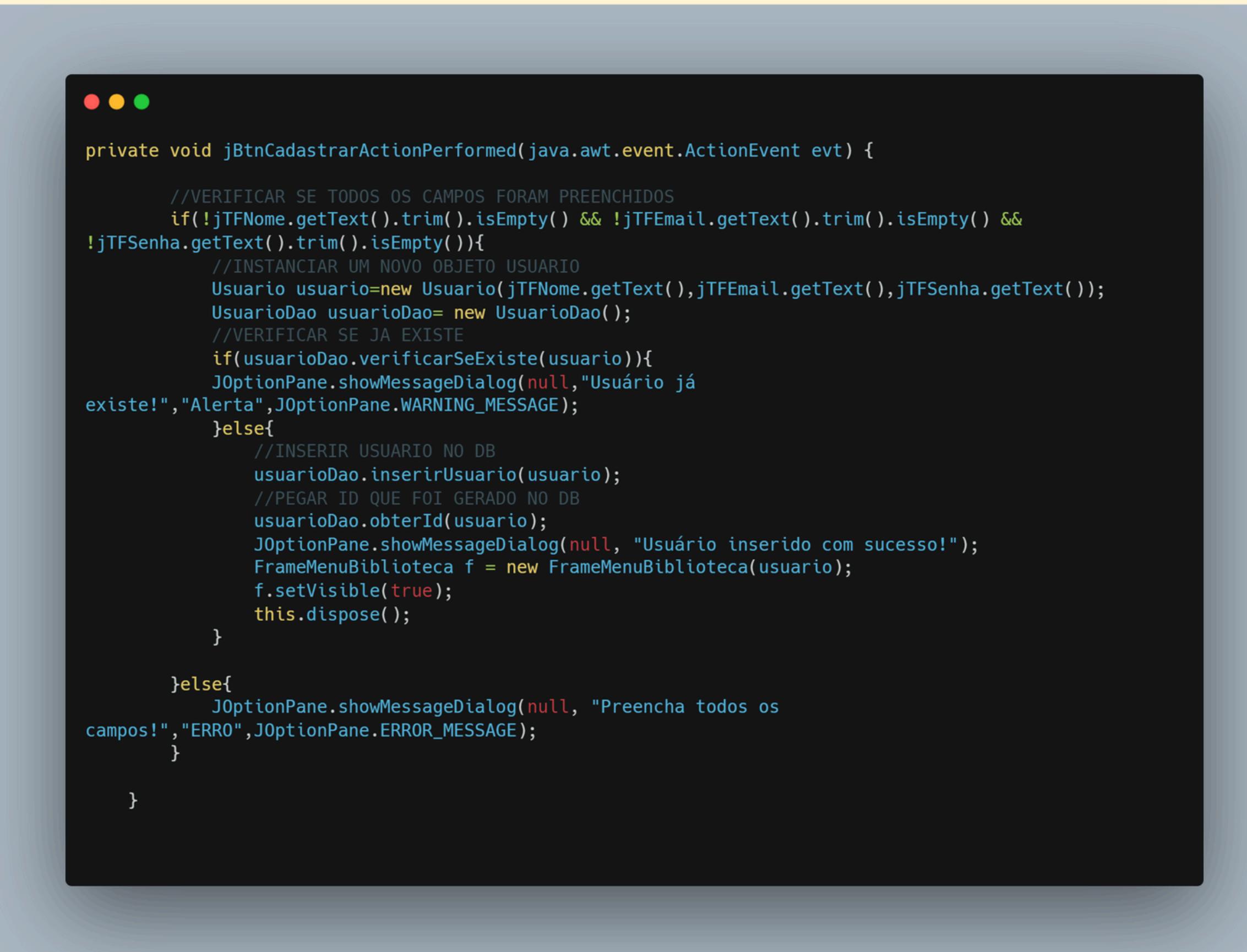
# MENU INICIAL



# CADASTRAR USUÁRIO



# BOTÃO CADASTRAR



The screenshot shows a Java Swing application window titled "BOTÃO CADASTRAR". The window contains a single text area displaying Java code. The code is a method named `jBtnCadastrarActionPerformed` which handles the action performed by a button. The code performs several tasks: it checks if three text fields (jTFNome, jTEmail, jTFSenha) are empty; if they are not empty, it creates a new `Usuario` object with the values from the text fields, and a `UsuarioDao` object. It then checks if the user already exists in the database. If the user exists, it shows an error message. If the user does not exist, it inserts the user into the database, gets the generated ID, shows a success message, creates a new frame (`FrameMenuBiblioteca`) with the user data, sets it visible, and then disposes of the current window. If any field is empty, it shows an error message asking to fill all fields.

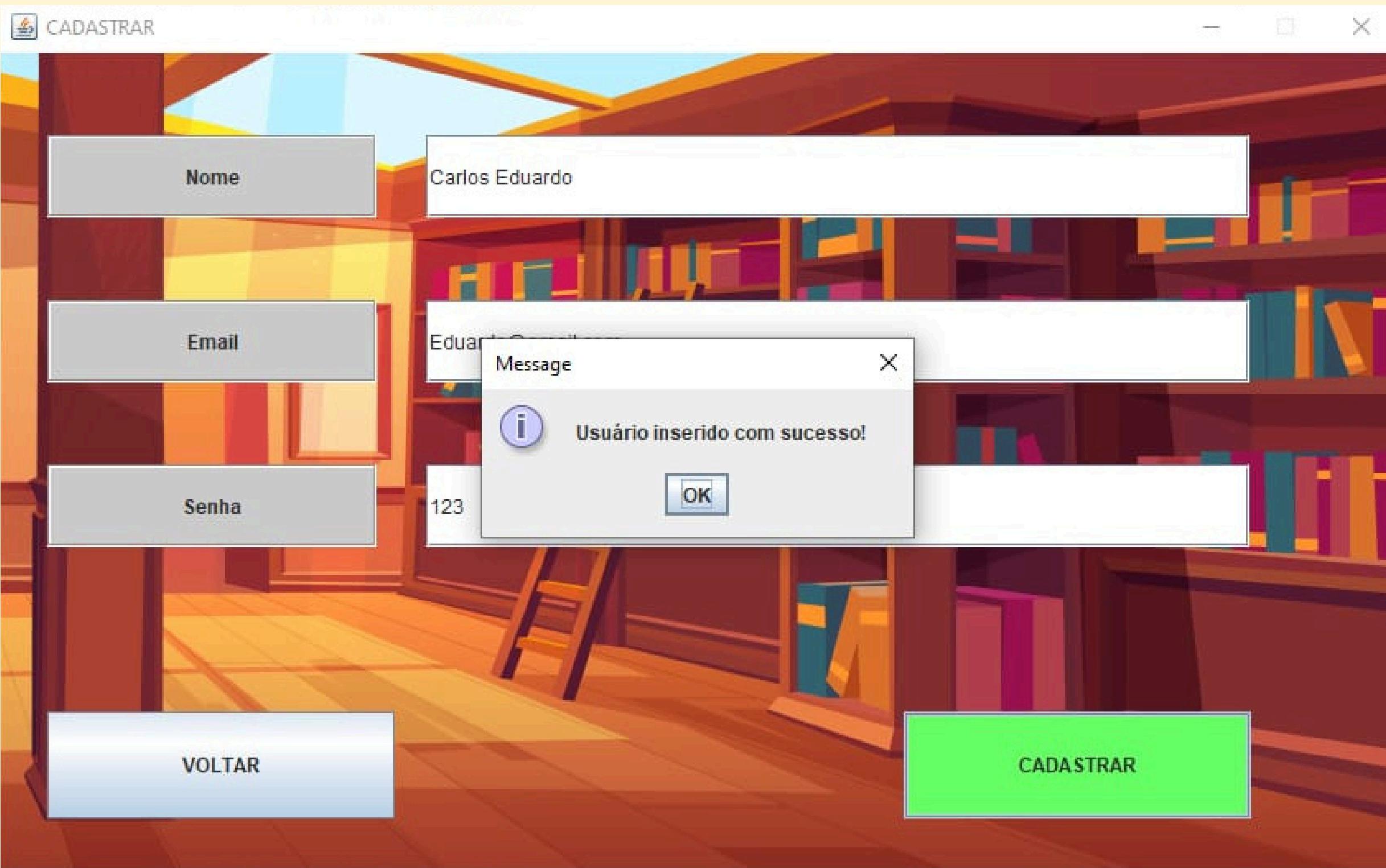
```
private void jBtnCadastrarActionPerformed(java.awt.event.ActionEvent evt) {  
    //VERIFICAR SE TODOS OS CAMPOS FORAM PREENCHIDOS  
    if(!jTFNome.getText().trim().isEmpty() && !jTEmail.getText().trim().isEmpty() &&  
    !jTFSenha.getText().trim().isEmpty()){  
        //INSTANCIAR UM NOVO OBJETO USUARIO  
        Usuario usuario=new Usuario(jTFNome.getText(),jTEmail.getText(),jTFSenha.getText());  
        UsuarioDao usuarioDao= new UsuarioDao();  
        //VERIFICAR SE JA EXISTE  
        if(usuarioDao.verificarSeExiste(usuario)){  
            JOptionPane.showMessageDialog(null,"Usuário já  
existe!","Alerta",JOptionPane.WARNING_MESSAGE);  
        }else{  
            //INSERIR USUARIO NO DB  
            usuarioDao.inserirUsuario(usuario);  
            //PEGAR ID QUE FOI GERADO NO DB  
            usuarioDao.obterId(usuario);  
            JOptionPane.showMessageDialog(null, "Usuário inserido com sucesso!");  
            FrameMenuBiblioteca f = new FrameMenuBiblioteca(usuario);  
            f.setVisible(true);  
            this.dispose();  
        }  
    }else{  
        JOptionPane.showMessageDialog(null, "Preencha todos os  
campos!","ERRO",JOptionPane.ERROR_MESSAGE);  
    }  
}
```

# MÉTODOS CADASTRAR

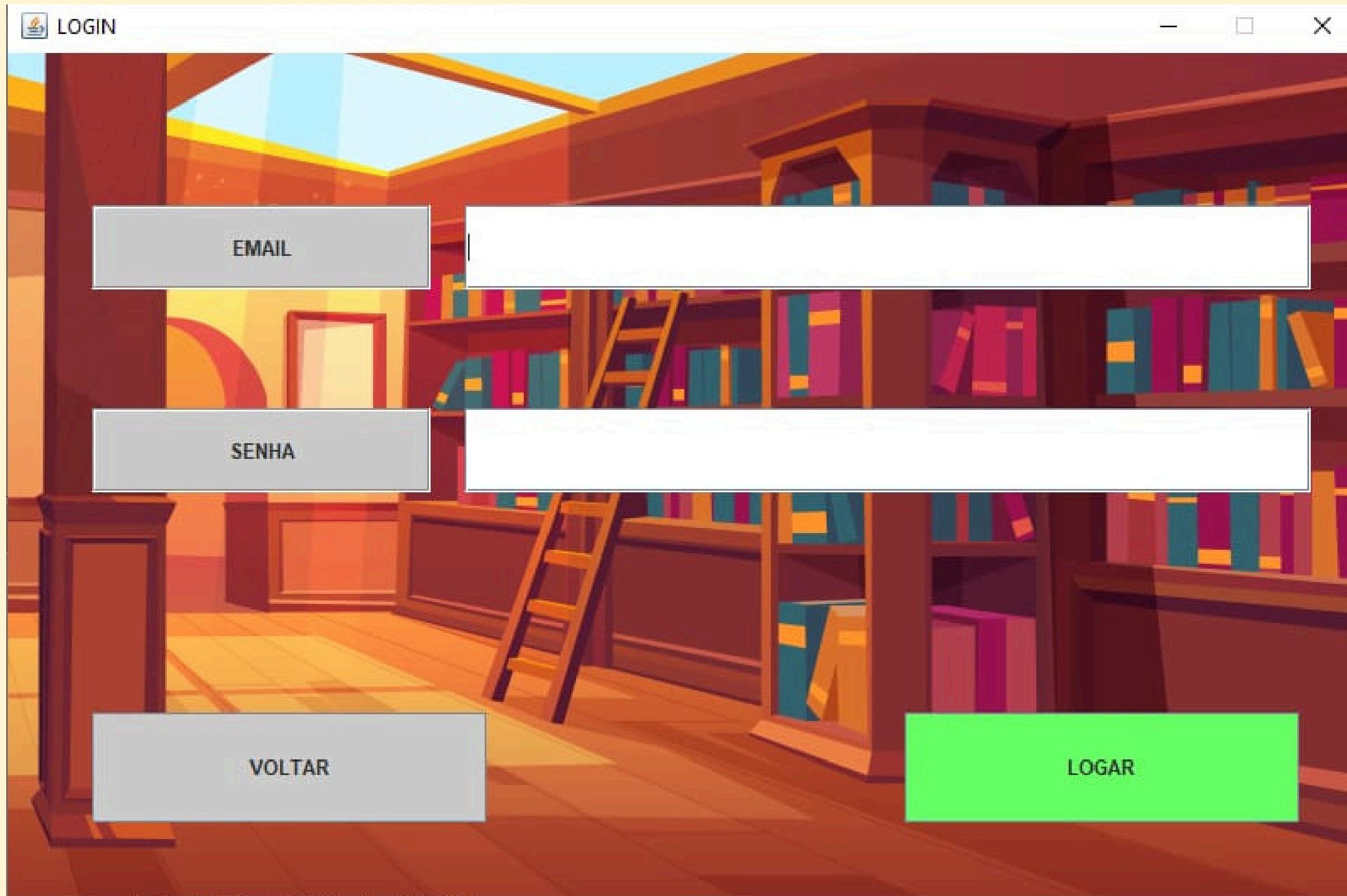
```
public static boolean verificarSeExiste(Usuario usuario) {  
  
    String sql = "SELECT COUNT(*) FROM usuarios WHERE nome=? AND email=? AND senha=?";  
    con=new Conexao().obterConexao();  
  
    try {  
        PreparedStatement pstm = con.prepareStatement(sql);  
  
        pstm.setString(1, usuario.getNome());  
        pstm.setString(2, usuario.getEmail());  
        pstm.setString(3, usuario.getSenha());  
  
        ResultSet rs = pstm.executeQuery();  
        if(rs.next()){  
            int count = rs.getInt(1);  
            return count>0;  
        }  
  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null, "Erro ao verificar se a pessoa já existe: " + e);  
    }return false;  
}
```

```
public void inserirUsuario(Usuario usuario) {  
    String sql = "INSERT INTO usuarios (nome,email,senha) values(?,?,?)";  
    con = new Conexao().obterConexao();  
    try {  
        PreparedStatement pstm = con.prepareStatement(sql);  
        pstm.setString(1, usuario.getNome());  
        pstm.setString(2, usuario.getEmail());  
        pstm.setString(3, usuario.getSenha());  
        pstm.executeUpdate();  
        pstm.close();  
        con.close();  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null, "Erro ao inserir usuário, Erro: " + e);  
    }  
  
    }public static void obterId(Usuario usuario){  
    con=new Conexao().obterConexao();  
    String sql = "SELECT id FROM usuarios WHERE email=? AND senha=?";  
    try {  
        PreparedStatement pstm = con.prepareStatement(sql);  
        pstm.setString(1, usuario.getEmail());  
        pstm.setString(2, usuario.getSenha());  
        ResultSet rs = pstm.executeQuery();  
        while(rs.next()){  
            int id=rs.getInt("id");  
            usuario.setId(id);  
        }  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog(null, "Erro ao obter id da pessoa: " + e);  
    }  
}
```

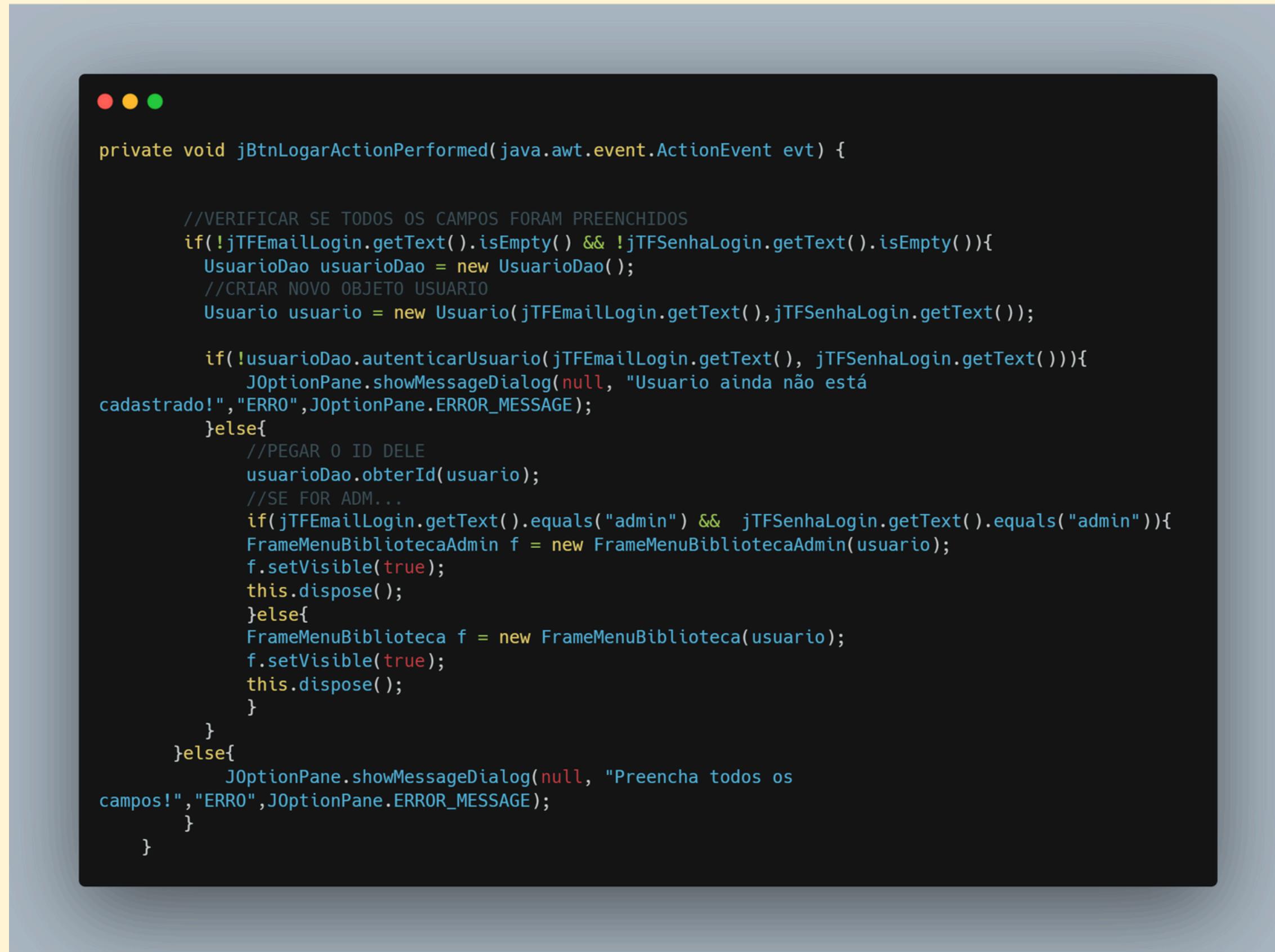
# USUÁRIO CADASTRADO



# LOGIN



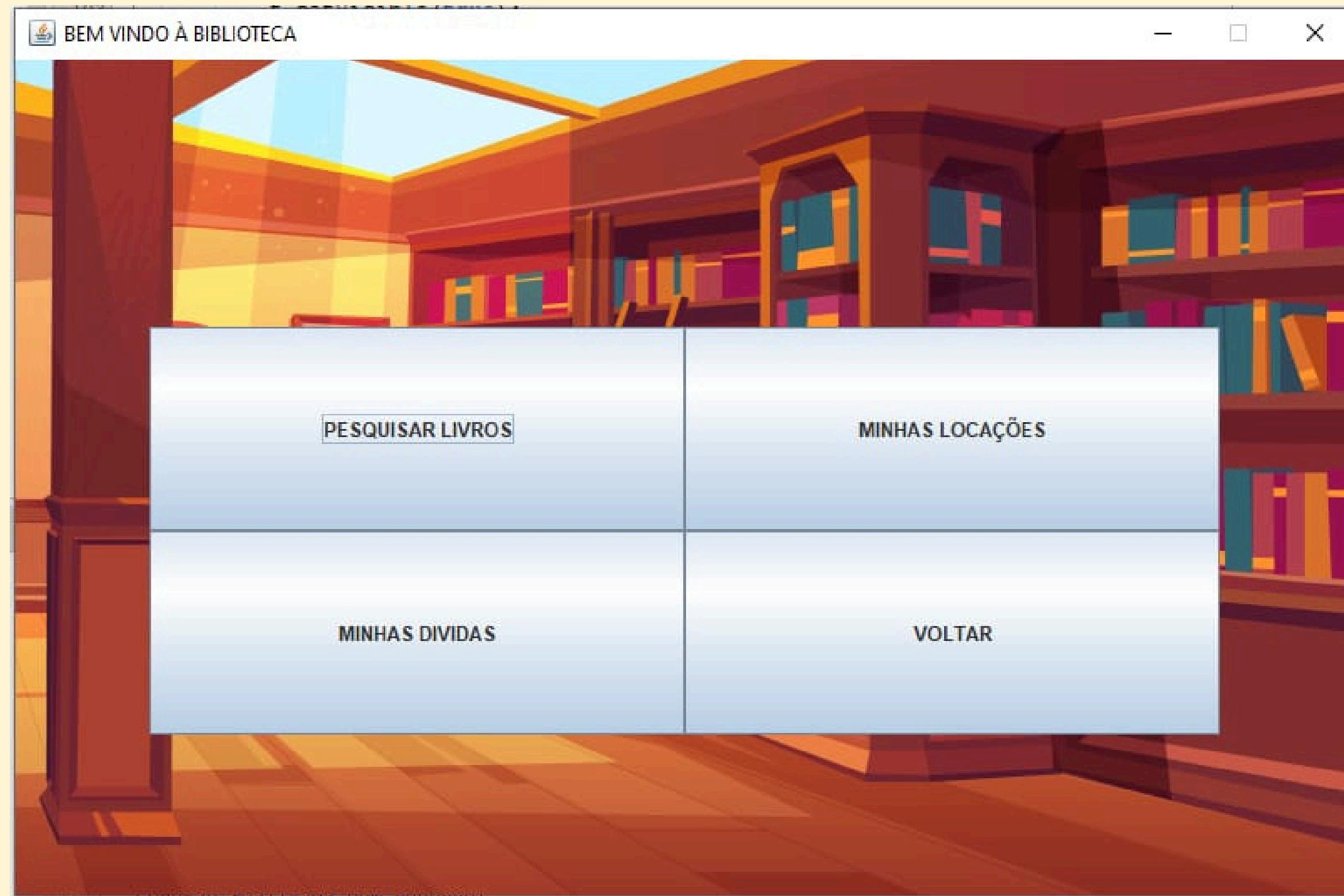
# BOTÃO LOGAR



A screenshot of a Java Swing application window titled "BOTÃO LOGAR". The window has a dark theme with red, yellow, and green window control buttons at the top left. The main area contains the following Java code:

```
private void jBtnLogarActionPerformed(java.awt.event.ActionEvent evt) {  
  
    //VERIFICAR SE TODOS OS CAMPOS FORAM PREENCHIDOS  
    if(!jTFEmailLogin.getText().isEmpty() && !jTFSenhaLogin.getText().isEmpty()){  
        UsuarioDao usuarioDao = new UsuarioDao();  
        //CRIAR NOVO OBJETO USUÁRIO  
        Usuario usuario = new Usuario(jTFEmailLogin.getText(),jTFSenhaLogin.getText());  
  
        if(!usuarioDao.autenticarUsuario(jTFEmailLogin.getText(), jTFSenhaLogin.getText())){  
            JOptionPane.showMessageDialog(null, "Usuario ainda não está  
cadastrado!", "ERRO", JOptionPane.ERROR_MESSAGE);  
        }else{  
            //PEGAR O ID DELE  
            usuarioDao.obterId(usuario);  
            //SE FOR ADM...  
            if(jTFEmailLogin.getText().equals("admin") && jTFSenhaLogin.getText().equals("admin")){  
                FrameMenuBibliotecaAdmin f = new FrameMenuBibliotecaAdmin(usuario);  
                f.setVisible(true);  
                this.dispose();  
            }else{  
                FrameMenuBiblioteca f = new FrameMenuBiblioteca(usuario);  
                f.setVisible(true);  
                this.dispose();  
            }  
        }  
    }else{  
        JOptionPane.showMessageDialog(null, "Preencha todos os  
campos!", "ERRO", JOptionPane.ERROR_MESSAGE);  
    }  
}
```

# MENU BIBLIOTECA USUÁRIO



# AGREGAÇÃO



The screenshot shows a Java code editor window with a dark theme. At the top left are three small colored circles (red, yellow, green). The code itself is written in Java and contains several event-handling methods:

```
private void jBtnPesquisarLivrosActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jBtnPesquisarLivrosActionPerformed
    FramePesquisarLivros f = new FramePesquisarLivros(usuario);
    f.setVisible(true);
    this.dispose();
} //GEN-LAST:event_jBtnPesquisarLivrosActionPerformed

private void jBtnVoltarActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jBtnVoltarActionPerformed
    FrameLogin f = new FrameLogin();
    f.setVisible(true);
    this.dispose();
} //GEN-LAST:event_jBtnVoltarActionPerformed

private void jBtnLocacoesActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jBtnLocacoesActionPerformed
    FrameLocacoes f = new FrameLocacoes(usuario);
    f.setVisible(true);
    this.dispose();
} //GEN-LAST:event_jBtnLocacoesActionPerformed

private void jBtnDividasActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jBtnDividasActionPerformed
    FrameDividas f = new FrameDividas(usuario);
    f.setVisible(true);
    this.dispose();
}
```

```
private void jBtnPesquisarLivrosActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jBtnPesquisarLivrosActionPerformed
    FramePesquisarLivros f = new FramePesquisarLivros(usuario);
    f.setVisible(true);
    this.dispose();
}

public class FramePesquisarLivros extends javax.swing.JFrame {

    private static Usuario usuario;

    public FramePesquisarLivros(Usuario usuario) {
        this.usuario = usuario;
        initComponents();
        listarTabela();
        redimensionarImagem();
    }
}
```

# PESQUISAR LIVROS



The window title is "PESQUISAR LIVROS". It features two buttons: "ALUGAR" on the left and "VOLTAR" on the right. Below these buttons is a table with the following data:

Id	Titulo	Autor	Genero	Disponiveis
1	O Senhor dos Anéis de Fogo e de Gelo	J.R.R. Tolkien	Fantasia	4
2	1984	George Orwell	Ficção Distópica	2
3	Dom Quixote	Miguel de Cervantes	Clássico	3
4	Orgulho e Preconceito	Jane Austen	Romance	3
5	O Pequeno Príncipe	Antoine de Saint-Exupéry	Infantil	6
6	A Revolução dos Bichos	George Orwell	Fábula	7
7	Cem Anos de Solidão	Gabriel García Márquez	Realismo Mágico	3
8	Moby Dick	Herman Melville	Aventura	4
9	O Grande Gatsby	F. Scott Fitzgerald	Romance	2
10	Crime e Castigo	Fyodor Dostoevsky	Drama	5

# BOTÃO ALUGAR



The screenshot shows a Java code editor window with a dark theme. The code is written in Java and handles the logic for the 'Alugar' (Rent) button. It includes comments in Portuguese explaining the steps: getting the selected row from a table, checking if it's valid, getting the book ID, verifying if the book is available, creating a new rental object, and updating the database. It also checks if the user has already rented the book and displays messages based on the outcome.

```
private void jBtnAlugarActionPerformed(java.awt.event.ActionEvent evt) {  
    //PEGAR O NUMERO DA LINHA  
    int setar = jTableLivros.getSelectedRow();  
    //VERIFICAR SE TEM UMA LINHA SELECIONADA  
    if (setar != -1) {  
        String qtdString = jTableLivros.getModel().getValueAt(setar, 4).toString();  
        int qtdInt = Integer.parseInt(qtdString);  
        //VERIFICAR SE TEM LIVRO DISPONIVEL  
        if (qtdInt > 0) {  
            String idLivroString = jTableLivros.getModel().getValueAt(setar, 0).toString();  
            int idLivroInt = Integer.parseInt(idLivroString);  
  
            //NOVO ALUGUEL  
            Aluguel aluguel = new Aluguel(usuario.getId(), idLivroInt);  
            AluguelDao aluguelDao = new AluguelDao();  
  
            // Verifica se o usuário ja alugou o livro  
            boolean jaAlugado = aluguelDao.verificarAluguelExistente(usuario.getId(), idLivroInt);  
  
            if (!jaAlugado) {  
                aluguelDao.inserirAluguel(aluguel);  
                aluguelDao.atualizarAluguelDisponiveisDecrementar(aluguel); // Atualiza a  
disponibilidade do livro  
  
                limparTabela(); // Limpa a tabela antes de listar novamente  
                listarTabela(); // Lista os livros atualizados  
  
                JOptionPane.showMessageDialog(null, "Alugado com sucesso!", "AVISO",  
JOptionPane.INFORMATION_MESSAGE);  
            } else {  
                JOptionPane.showMessageDialog(null, "Este livro já foi alugado por você!", "AVISO",  
JOptionPane.WARNING_MESSAGE);  
            }  
            } else {  
                JOptionPane.showMessageDialog(null, "Este livro está indisponível no momento!", "AVISO",  
JOptionPane.INFORMATION_MESSAGE);  
            }  
            } else {  
                JOptionPane.showMessageDialog(null, "Selecione um livro!", "ERRO", JOptionPane.ERROR_MESSAGE);  
            }  
    }  
}
```

# MODEL ALUGUEL

```
● ● ●

public class Aluguel {
    private int id;
    private int idUsuario;
    private int idLivro;
    private LocalDate dataAluguel;
    private LocalDate dataDevolucao;

    public Aluguel(int id, int idUsuario, int idLivro) {
        this.id = id;
        this.idUsuario = idUsuario;
        this.idLivro = idLivro;
        this.dataAluguel = LocalDate.now(); //pega data atual
        this.dataDevolucao = dataAluguel.plusDays(15); //data atual mais 15 dias
    }
    public Aluguel(int idUsuario, int idLivro) {
        this.idUsuario = idUsuario;
        this.idLivro = idLivro;
        this.dataAluguel = LocalDate.now();
        this.dataDevolucao = dataAluguel.plusDays(15);
    }
}
```

# MÉTODOS ALUGUEL



```
public static boolean verificarAluguelExistente(int idUsuario, int idLivro) {
    String sql = "SELECT COUNT(*) FROM alugueis WHERE id_usuario = ? AND id_livros = ?";
    con = new Conexao().obterConexao();
    boolean existe = false;

    try {
        PreparedStatement pstm = con.prepareStatement(sql);
        pstm.setInt(1, idUsuario);
        pstm.setInt(2, idLivro);
        ResultSet rs = pstm.executeQuery();

        if (rs.next()) {
            int count = rs.getInt(1); // Obtém o número de registros encontrados
            existe = (count > 0); // Se count for maior que 0, o livro já está alugado
        }

        rs.close();
        pstm.close();
        con.close();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erro ao verificar aluguel existente, Erro: " + e,
        "Erro", JOptionPane.ERROR_MESSAGE);
    }

    return existe;
}

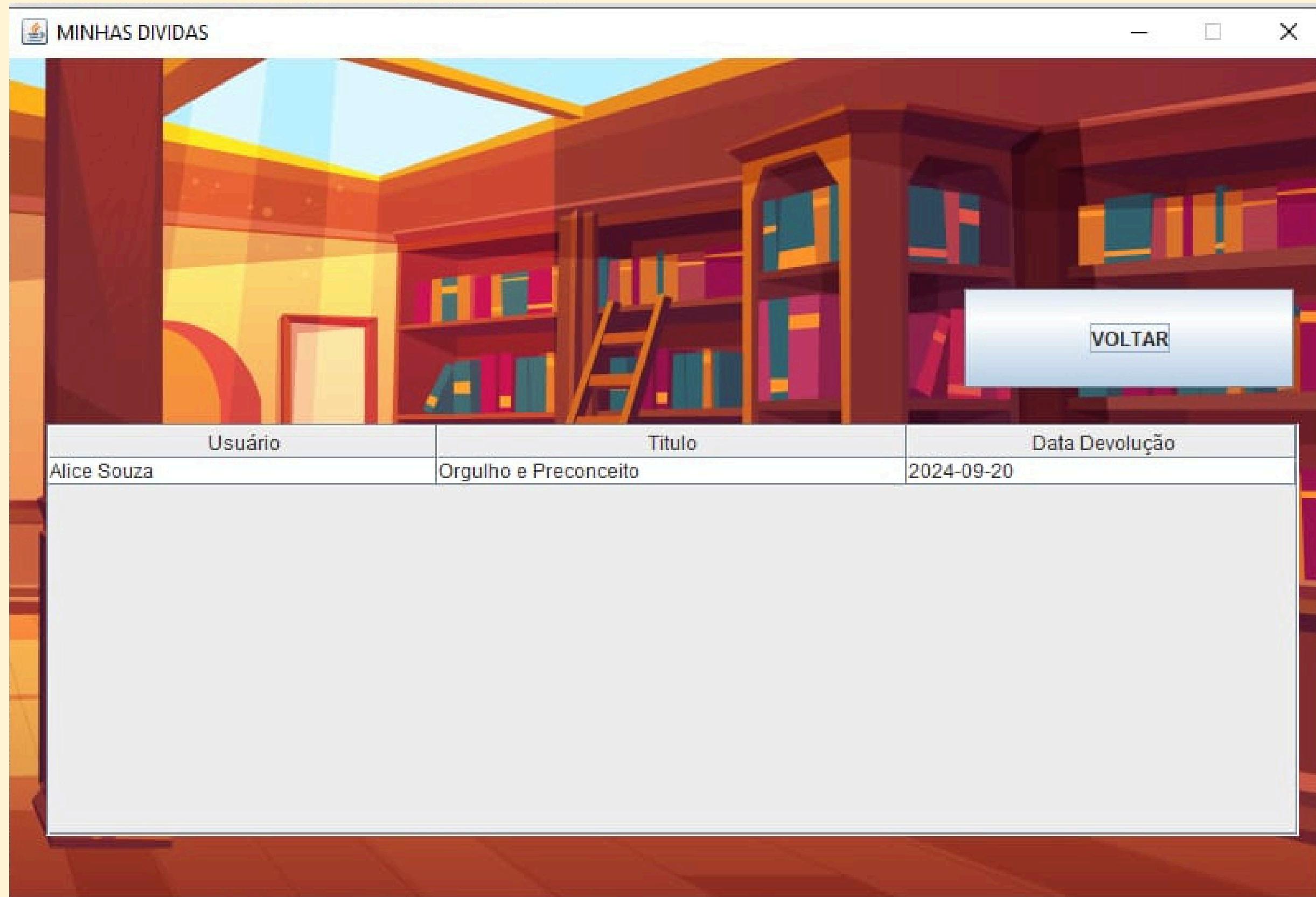
public static void atualizarAluguelDisponiveisDecrementar(Aluguel aluguel) {
    String sql = "UPDATE livros SET disponiveis = disponiveis - 1 WHERE id = ?";

    //colocar metodo para buscar livro por id
    con = new Conexao().obterConexao();

    try {
        PreparedStatement pstm = con.prepareStatement(sql);
        pstm.setInt(1, aluguel.getIdLivro());
        pstm.executeUpdate();

        pstm.close();
        con.close();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erro ao atualizar aluguel, Erro: " + e, "Erro",
        JOptionPane.ERROR_MESSAGE);
    }
}
```

# DÍVIDAS USUÁRIO



# LISTAR ALUGUEIS ATRASADOS



The screenshot shows a Java code editor with a dark theme. The code is written in Java and performs the following tasks:

- It defines a method `listarAlugueisTabela()` that retrieves delayed rentals from a database and displays them in a table.
- It uses a `DefaultTableModel` to set up the table columns and their widths.
- It clears the table's row count before adding new data.
- It iterates through the list of delayed rentals and adds each one as a new row in the table model.
- It contains a static method `listarAlugueisAtrasadosUsuario(Usuario usuario)` which performs a database query to find rentals where the return date has passed the current date.
- The query uses JOIN clauses to link users, books, and rentals tables, and filters by user ID and rental status.
- It uses a `PreparedStatement` to execute the query, passing the user ID as a parameter.
- It processes the results from the database query, extracting user name, book title, and return date, and adds these details to the list of delayed rentals.
- Finally, it closes the database connection and returns the list of delayed rentals.

# LOCAÇÕES USUÁRIO

MINHAS LOCAÇÕES

DEVOLVER

VOLTAR

id	Titulo	Autor	Genero
39	1984	George Orwell	Distopia
41	O Senhor dos Anéis	J.R.R. Tolkien	Fantasia
42	Orgulho e Preconceito	Jane Austen	Romance
43	Cem Anos de Solidão	Gabriel Garcia Marquez	Realismo Mágico
44	A Revolução dos Bichos	George Orwell	Fábula
38	Dom Quixote	Miguel de Cervantes	Clássico

# BOTÃO DEVOLVER



The screenshot shows a Java Swing application window titled "BOTÃO DEVOLVER". The window has a dark theme with red, yellow, and green close/minimize/maximize buttons at the top-left. The main area contains the following Java code:

```
private void jBtnDevolverActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jBtnDevolverActionPerformed
    //SELECCIONA O NUMERO DA LINHA
    int setar = jTableLocacoes.getSelectedRow();
    if (setar != -1) {
        //PEGA ID DO LIVRO
        String idLivroString = jTableLocacoes.getModel().getValueAt(setar, 0).toString();
        int idLivroInt = Integer.parseInt(idLivroString);

        Aluguel aluguel = new Aluguel(usuario.getId(), idLivroInt);
        AluguelDao aluguelDao = new AluguelDao();
        //DELETO O ALUGUEL, INCREMENTO A QTD DISPONIVEIS DO LIVRO E LISTO DE NOVO A LISTA DE
        LOCACAO
        aluguelDao.deletarAluguel(aluguel);
        JOptionPane.showMessageDialog(null, "Livro devolvido com sucesso!", "AVISO",
        JOptionPane.INFORMATION_MESSAGE);
        aluguelDao.atualizarAluguelDisponiveisIncrementar(aluguel);
        limparTabela();
        listarLocacoes();
    } else{
        JOptionPane.showMessageDialog(null, "Selecione uma linha!", "AVISO",
        JOptionPane.WARNING_MESSAGE);
    }
}
```

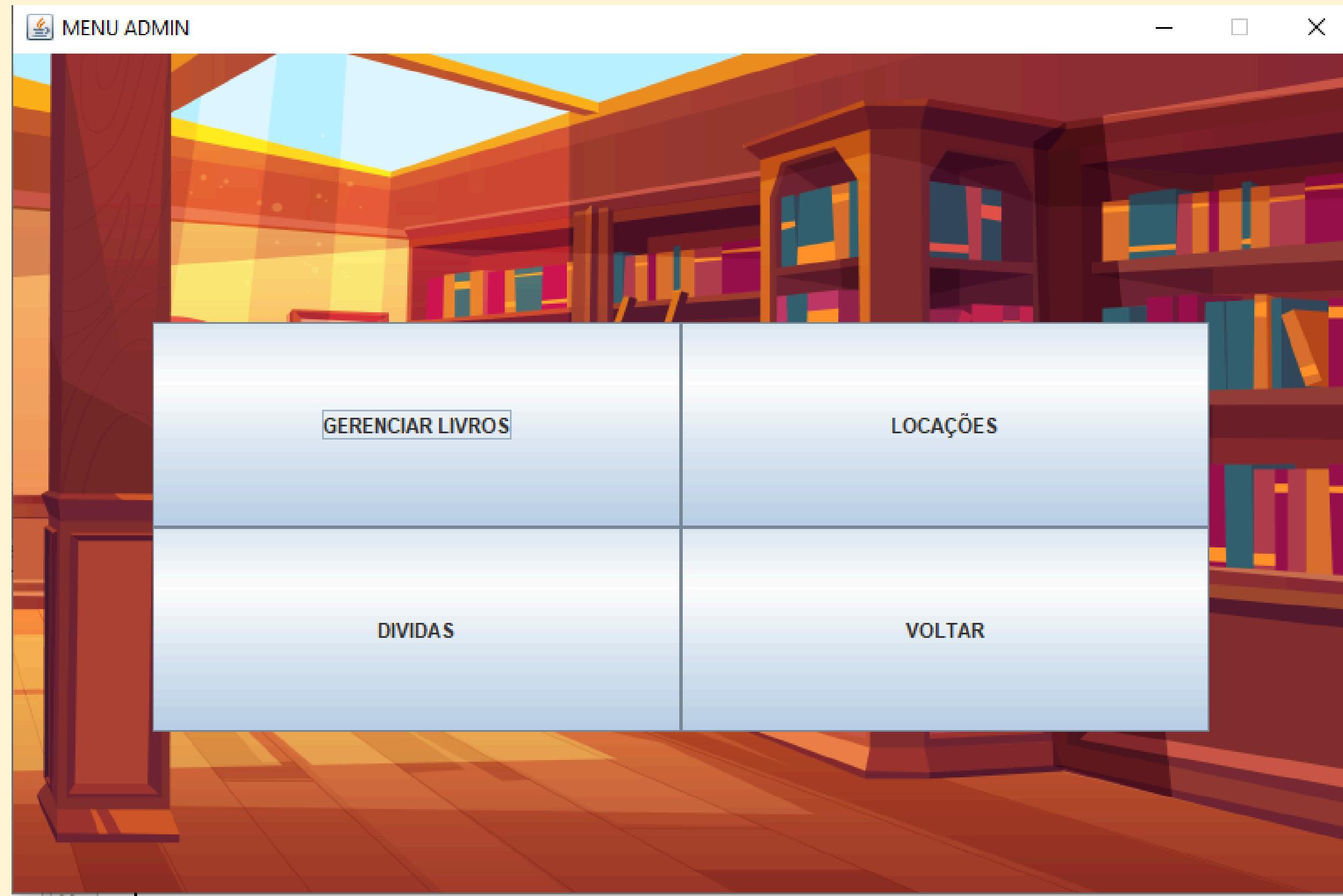
# MÉTODOS DEVOLVER



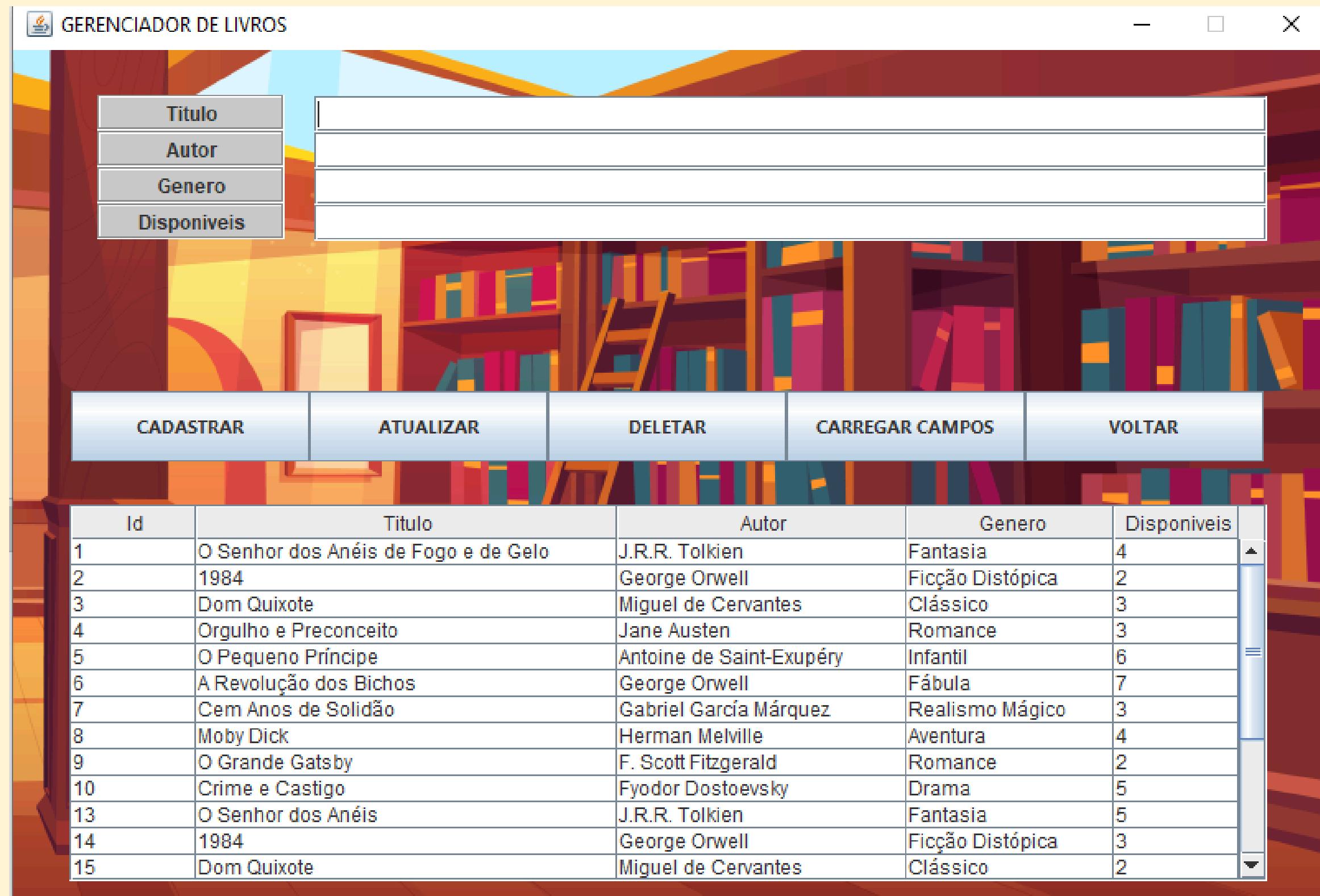
```
public static void deletarAluguel(Aluguel aluguel) {
    String sql = "DELETE FROM alugueis WHERE id_usuario=? AND id_livros=? ";
    con = new Conexao().obterConexao();
    try {
        PreparedStatement pstm = con.prepareStatement(sql);
        pstm.setInt(1, aluguel.getIdUsuario());
        pstm.setInt(2, aluguel.getIdLivro());
        pstm.executeUpdate();
        pstm.close();
        con.close();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erro ao deletar aluguel, Erro: " + e, "Erro",
JOptionPane.ERROR_MESSAGE);
    }
}

public static void atualizarAluguelDisponiveisIncrementar(Aluguel aluguel) {
    String sql = "UPDATE livros SET disponiveis = disponiveis + 1 WHERE id = ?";
    //colocar metodo para buscar livro por id
    con = new Conexao().obterConexao();
    try {
        PreparedStatement pstm = con.prepareStatement(sql);
        pstm.setInt(1, aluguel.getIdLivro());
        pstm.executeUpdate();
        pstm.close();
        con.close();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erro ao atualizar aluguel, Erro: " + e, "Erro",
JOptionPane.ERROR_MESSAGE);
    }
}
```

# MENU BIBLIOTECA ADMIN



# GERENCIADOR DE LIVROS



# BOTÃO CADASTRAR LIVRO

```
private void jBtnCadastrarActionPerformed(java.awt.event.ActionEvent evt) {  
    cadastrarlivro();  
    listarTabela();  
    limparCampos();  
}  
private void cadastrarlivro() {  
    String titulo, autor, genero, disponiveis;  
  
    titulo = jTFTitulo.getText();  
    autor = jTFAutor.getText();  
    genero = jTFGenero.getText();  
    disponiveis = jTFDisponiveis.getText();  
  
    // Verificando se algum campo está vazio  
    if (titulo.trim().isEmpty() || autor.trim().isEmpty() || genero.trim().isEmpty() ||  
disponiveis.trim().isEmpty()) {  
        JOptionPane.showMessageDialog(null, "Todos os campos devem ser preenchidos!", "Erro",  
JOptionPane.ERROR_MESSAGE);  
        return;  
    }  
  
    try {  
        // Convertendo a string disponiveis para int  
        int qtdDisponiveis = Integer.parseInt(disponiveis);  
  
        // Criando o objeto Livro  
        Livro livro = new Livro(titulo, autor, genero, qtdDisponiveis);  
  
        // Inserindo no banco de dados  
        LivroDao livroDao = new LivroDao();  
        //verificar se livro ja nao esta cadastrado  
        if (!livroDao.verificarSeLivroExiste(livro)) {  
            livroDao.inserirLivro(livro);  
        }else{  
            JOptionPane.showMessageDialog(null, "Este livro já está cadastrado!", "AVISO",  
JOptionPane.WARNING_MESSAGE);  
        }  
  
    } catch (NumberFormatException e) {  
        // Tratamento caso o valor de "disponiveis" não seja um número válido  
        JOptionPane.showMessageDialog(null, "O campo 'Disponíveis' deve conter um número válido!",  
"Erro", JOptionPane.ERROR_MESSAGE);  
    }  
}
```

# BOTÃO ATUALIZAR LIVRO

```
private void jBtnAtualizarActionPerformed(java.awt.event.ActionEvent evt) {  
    atualizar();  
    listarTabela();  
    limparCampos();  
}  
private void atualizar() {  
    int setar = jTableLivros.getSelectedRow();  
    if (setar != -1) {  
        String idLivroString = jTableLivros.getModel().getValueAt(setar, 0).toString();  
        int idLivroInt = Integer.parseInt(idLivroString);  
  
        String titulo = jTFTitulo.getText();  
        String autor = jTFAutor.getText();  
        String genero = jTFGenero.getText();  
        String disponiveisString = jTFDisponiveis.getText();  
  
        // Verificar se os campos estão preenchidos corretamente  
        if (titulo.trim().isEmpty() || autor.trim().isEmpty() || genero.trim().isEmpty() ||  
            disponiveisString.trim().isEmpty()) {  
            JOptionPane.showMessageDialog(null, "Todos os campos devem ser preenchidos!", "Erro",  
                JOptionPane.ERROR_MESSAGE);  
            return;  
        }  
  
        try {  
            int disponiveis = Integer.parseInt(disponiveisString);  
  
            // Criando o objeto Livro  
            Livro livro = new Livro(idLivroInt, titulo, autor, genero, disponiveis);  
  
            // Atualizando no banco de dados  
            LivroDao livroDao = new LivroDao();  
            livroDao.atualizarLivro(livro);  
  
            // Atualizar a tabela após a alteração  
            listarTabela();  
            limparCampos();  
  
        } catch (NumberFormatException e) {  
            JOptionPane.showMessageDialog(null, "O campo 'Disponíveis' deve conter um número  
válido!", "Erro", JOptionPane.ERROR_MESSAGE);  
        }  
  
    } else {  
        JOptionPane.showMessageDialog(null, "Selecione uma linha!", "AVISO",  
            JOptionPane.WARNING_MESSAGE);  
    }  
}
```

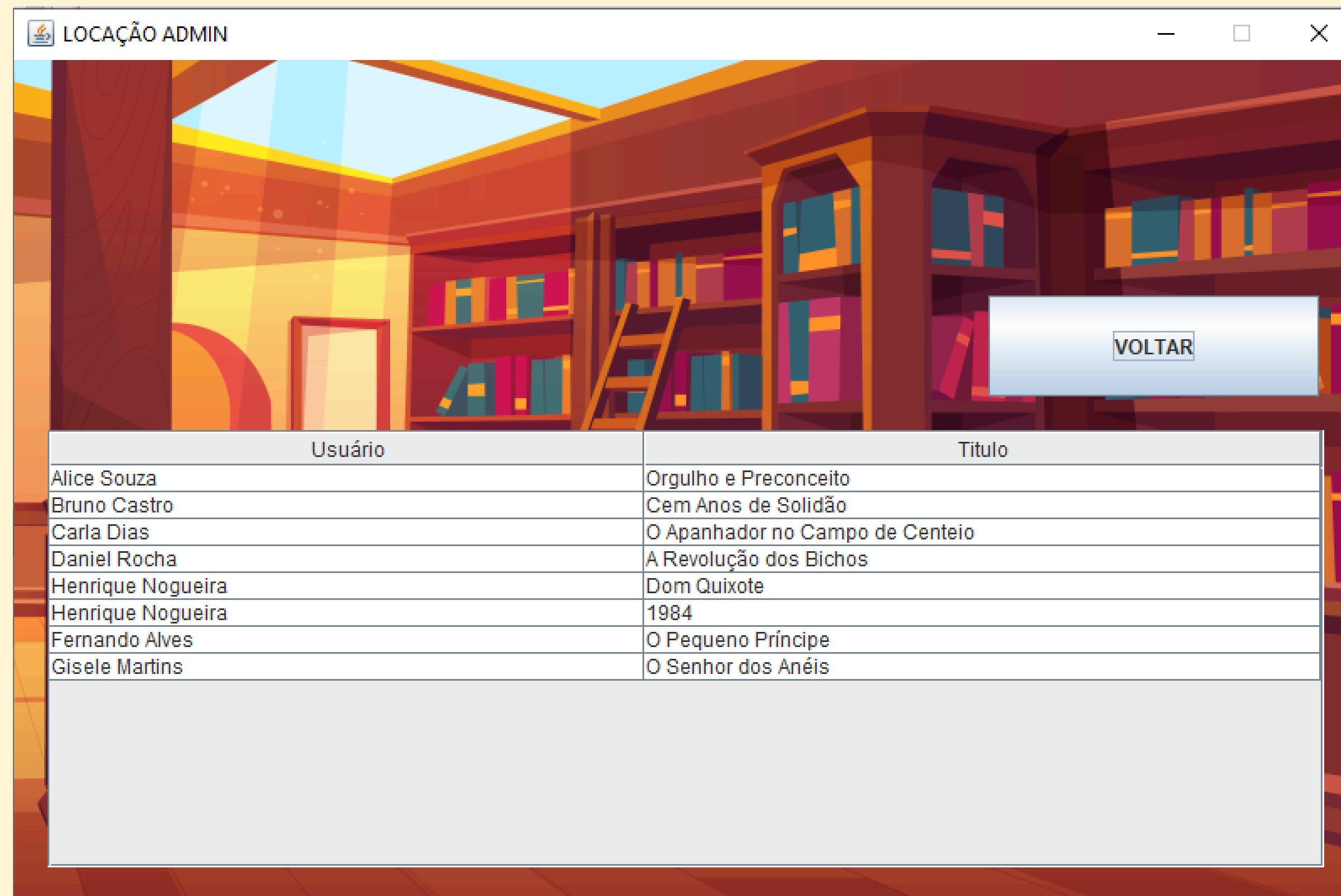
# BOTÃO DELETAR LIVRO



The screenshot shows a Java Swing application window titled "BOTÃO DELETAR LIVRO". The window has a dark theme with red, yellow, and green window control buttons at the top. The main area contains the following Java code:

```
private void jBtnDeletarActionPerformed(java.awt.event.ActionEvent evt) {  
  
    excluirLivro();  
    listarTabela();  
    limparCampos();  
}  
private void excluirLivro() {  
    int setar = jTableLivros.getSelectedRow();  
    if (setar != -1) {  
        String idLivroString = jTableLivros.getModel().getValueAt(setar, 0).toString();  
        int idLivroInt = Integer.parseInt(idLivroString);  
  
        Livro livro = new Livro();  
        livro.setId(idLivroInt);  
  
        LivroDao livroDao = new LivroDao();  
        livroDao.deletarLivroPorId(livro);  
        limparTabela();  
    } else {  
        JOptionPane.showMessageDialog(null, "Selecione uma linha!", "AVISO",  
        JOptionPane.WARNING_MESSAGE);  
    }  
}
```

# LOCAÇÕES ADMIN



The application window has a title bar "LOCAÇÃO ADMIN" with a small icon. It features a background illustration of a library interior with wooden bookshelves filled with books and a ladder leaning against one of them. In the foreground, there is a light blue rectangular button with the word "VOLTAR" (Back) in black capital letters.

Usuário	Titulo
Alice Souza	Orgulho e Preconceito
Bruno Castro	Cem Anos de Solidão
Carla Dias	O Apanhador no Campo de Centeio
Daniel Rocha	A Revolução dos Bichos
Henrique Nogueira	Dom Quixote
Henrique Nogueira	1984
Fernando Alves	O Pequeno Príncipe
Gisele Martins	O Senhor dos Anéis

# LISTAR TODOS ALUGUEIS

```
public void listarAlugueisTabela() {
    ArrayList<String[]> listaAlugueis = AluguelDao.listarAlugueisComDetalhes();
    DefaultTableModel dtm = (DefaultTableModel) jTableLocacoes.getModel();

    // Configura as colunas (Nome do Usuário, Título do Livro)
    jTableLocacoes.getColumnModel().getColumn(0).setPreferredWidth(150);
    jTableLocacoes.getColumnModel().getColumn(1).setPreferredWidth(200);

    // Limpa a tabela antes de adicionar os novos dados
    dtm.setRowCount(0);

    // Adiciona as linhas com nome do usuário e título do livro
    for (String[] aluguel : listaAlugueis) {
        dtm.addRow(aluguel);
    }
}

public static ArrayList<String[]> listarAlugueisComDetalhes() {
    con = new Conexao().obterConexao();
    ArrayList<String[]> listaAlugueis = new ArrayList<>();

    // Consulta que une as tabelas para obter nome do usuário e título do livro
    String sql = "SELECT u.nome AS nome_usuario, l.titulo AS titulo_livro
                 FROM alugueis a
                 JOIN usuarios u ON a.id_usuario = u.id
                 JOIN livros l ON a.id_livros = l.id";

    try {
        PreparedStatement pstmt = con.prepareStatement(sql);
        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {
            String nomeUsuario = rs.getString("nome_usuario");
            String tituloLivro = rs.getString("titulo_livro");

            // Adiciona os resultados em um array de Strings para cada linha
            String[] detalhesAluguel = {nomeUsuario, tituloLivro};
            listaAlugueis.add(detalhesAluguel);
        }

        rs.close();
        pstmt.close();
        con.close();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erro ao obter dados de aluguel: " + e, "ERRO",
JOptionPane.ERROR_MESSAGE);
    }
}

return listaAlugueis;
}
```

# DÍVIDAS ADMIN

 DIVIDAS ADMIN

[VOLTAR](#)

Usuário	Titulo	Data Devolução
Alice Souza	Orgulho e Preconceito	2024-09-20
Bruno Castro	Cem Anos de Solidão	2024-09-05
Carla Dias	O Apanhador no Campo de Centeio	2024-09-25
Daniel Rocha	A Revolução dos Bichos	2024-07-20
Henrique Nogueira	Dom Quixote	2024-07-20
Henrique Nogueira	1984	2024-09-15
Fernando Alves	O Pequeno Príncipe	2024-08-15
Gisele Martins	O Senhor dos Anéis	2024-10-15

# LISTAR TODAS AS DÍVIDAS

```
public void listarAlugueisTabela() {
    ArrayList<String[]> listaAlugueis = AluguelDao.listarAlugueisAtrasados();
    DefaultTableModel dtm = (DefaultTableModel) jTableDividas.getModel();

    // Configura as colunas (Nome do Usuário, Título do Livro)
    jTableDividas.getColumnModel().getColumn(0).setPreferredWidth(100);
    jTableDividas.getColumnModel().getColumn(1).setPreferredWidth(150);
    jTableDividas.getColumnModel().getColumn(2).setPreferredWidth(100);

    // Limpa a tabela antes de adicionar os novos dados
    dtm.setRowCount(0);

    // Adiciona as linhas com nome do usuário e título do livro
    for (String[] aluguel : listaAlugueis) {
        dtm.addRow(aluguel);
    }
}

public static ArrayList<String[]> listarAlugueisAtrasados() {
    con = new Conexao().obterConexao();
    ArrayList<String[]> listaAlugueisAtrasados = new ArrayList<>();

    // Consulta para pegar os usuários e livros cujas datas de devolução já passaram
    String sql = "SELECT u.nome AS nome_usuario, l.titulo AS titulo_livro, a.data_devolucao
                 FROM alugueis a
                 JOIN usuarios u ON a.id_usuario = u.id
                 JOIN livros l ON a.id_livros = l.id
                 WHERE a.data_devolucao < CURRENT_DATE"; // Verifica se a data de devolução já passou
    da data atual

    try {
        PreparedStatement pstm = con.prepareStatement(sql);
        ResultSet rs = pstm.executeQuery();

        while (rs.next()) {
            String nomeUsuario = rs.getString("nome_usuario");
            String tituloLivro = rs.getString("titulo_livro");
            String dataDevolucao = rs.getString("data_devolucao");

            // Adiciona os resultados em um array de Strings para cada linha
            String[] detalhesAluguelAtrasado = {nomeUsuario, tituloLivro, dataDevolucao};
            listaAlugueisAtrasados.add(detalhesAluguelAtrasado);
        }

        rs.close();
        pstm.close();
        con.close();
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Erro ao obter dados de aluguéis atrasados: " + e,
        "ERRO", JOptionPane.ERROR_MESSAGE);
    }
    return listaAlugueisAtrasados;
}
```

# OUTRAS FUNÇÕES

```
public void limparTabela() {
    DefaultTableModel dtm = (DefaultTableModel) jTableLocacoes.getModel();
    dtm.setRowCount(0); // Remove todas as linhas da tabela
}

private void redimensionarImagem() {
    // Carregar a imagem original
    ImageIcon icon = new ImageIcon(getClass().getResource("/imagens/1722.jpg"));

    // Redimensionar a imagem para caber no container
    Image img = icon.getImage().getScaledInstance(jPanel1.getWidth(), jPanel1.getHeight(),
Image.SCALE_SMOOTH);

    // Definir a imagem redimensionada
    jLabelImagen.setIcon(new ImageIcon(img));
}
```

# **MEMBROS DO GRUPO**

**CARLOS EDUARDO MARTINS PENNA**  
**VINICIUS DE SOUZA STUCHI**  
**DAVI FERNANDO SOUSA DIAS**

**RA:G865421**  
**RA:N078GF1**  
**RA:R0014C7**