/** * Esta classe representa a visualização de listagem de livros. * O código-fonte foi descompilado de um arquivo .class usando o decompilador FernFlower. * * @author [Seu Nome] * @version 1.0 * @since 2024-06-16 *// Source code is decompiled from a .class file using FernFlower decompiler. package view; import java.awt.BorderLayout; import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import javax.swing.JOptionPane; import javax.swing.JPanel; import javax.swing.JScrollPane; import javax.swing.JTable; import javax.swing.table.DefaultTableModel; import model.ConnectionFactory; public class ListarLivrosView extends JPanel { private JTable tabelaLivros; private DefaultTableModel modeloTabela; /** * Cria uma instância da visualização de listagem de livros. */ public ListarLivrosView() { this.setLayout(new BorderLayout()); String[] colunas = new String[]{"ID", "T\u00edtulo", "Autor", "G\u00eanero", "Quantidade", "Status"}; Object[][] dados = new Object[0][]; this.modeloTabela = new 1(this, dados, colunas); this.tabelaLivros = new JTable(this.modeloTabela); this.tabelaLivros.setSelectionMode(0); this.add(new JScrollPane(this.tabelaLivros), "Center"); this.modeloTabela.addTableModelListener(new 2(this)); } /** * Carrega os livros do banco de dados e atualiza a tabela. */ public void carregarLivros() { try { Throwable var1 = null; Object var2 = null; try { Connection conn = ConnectionFactory.getConnection(); try { String sql = "SELECT * FROM livro"; PreparedStatement stmt = conn.prepareStatement(sql); ResultSet rs = stmt.executeQuery(); this.modeloTabela.setRowCount(0); while(rs.next()) { Object[] row = new Object[]{rs.getInt("id"), rs.getString("titulo"), rs.getString("autor"), rs.getString("genero"), rs.getInt("quantidade"), rs.getString("status")}; this.modeloTabela.addRow(row); } } finally { if (conn != null) { conn.close(); } } } catch (Throwable var15) { if (var1 == null) { var1 = var15; } else if (var1 != var15) { var1.addSuppressed(var15); } throw var1; } } catch (Exception var16) { var16.printStackTrace(); JOptionPane.showMessageDialog(this, "Erro ao carregar livros do banco de dados.", "Erro", 0); } } /** * Filtra os livros do banco de dados de acordo com o texto de pesquisa fornecido e atualiza a tabela. * * @param searchText O texto de pesquisa. */ public void filtrarLivros(String searchText) { try { Throwable var2 = null; Object var3 = null; try { Connection conn = ConnectionFactory.getConnection(); try { String sql = "SELECT * FROM livro WHERE titulo LIKE ? OR autor LIKE ? OR genero LIKE ?"; PreparedStatement stmt = conn.prepareStatement(sql); String queryText = "%" + searchText + "%"; stmt.setString(1, queryText); stmt.setString(2, queryText); stmt.setString(3, queryText); ResultSet rs = stmt.executeQuery(); this.modeloTabela.setRowCount(0); while(rs.next()) { Object[] row = new Object[]{rs.getInt("id"), rs.getString("titulo"), rs.getString("autor"), rs.getString("genero"), rs.getInt("quantidade"), rs.getString("status")}; this.modeloTabela.addRow(row); } } finally { if (conn != null) { conn.close(); } } } catch (Throwable var17) { if (var2 == null) { var2 = var17; } else if (var2 != var17) { var2.addSuppressed(var17); } throw var2; } } catch (Exception var18) { var18.printStackTrace(); JOptionPane.showMessageDialog(this, "Erro ao filtrar livros do banco de dados.", "Erro", 0); } } /** * Obtém o ID do livro selecionado na tabela. * * @return O ID do livro selecionado, ou -1 se nenhum livro estiver selecionado. */ public int getLivroIdSelecionado() { int linhaSelecionada = this.tabelaLivros.getSelectedRow(); return linhaSelecionada != -1 ? (Integer)this.modeloTabela.getValueAt(linhaSelecionada, 0) : -1; } /** * Obtém o status do livro selecionado na tabela. * * @return O status do livro selecionado, ou null se nenhum livro estiver selecionado. */ public String getStatusLivroSelecionado() { int linhaSelecionada = this.tabelaLivros.getSelectedRow(); return linhaSelecionada != -1 ? (String)this.modeloTabela.getValueAt(linhaSelecionada, 5) : null; } /** * Obtém o título do livro selecionado na tabela. * * @return O título do livro selecionado, ou null se nenhum livro estiver selecionado. */ public String getTituloLivroSelecionado() { int linhaSelecionada = this.tabelaLivros.getSelectedRow(); return linhaSelecionada != -1 ? (String)this.modeloTabela.getValueAt(linhaSelecionada, 1) : null; } }