Nome: Nickolas dos Santos Braga

PROJETO BIBLIOTECA VIRTUAL

Relatório – Princípios SOLID e Refatorações

Princípios SOLID Aplicados:

A seguir são descritos os princípios SOLID aplicados no projeto, as classes afetadas e a justificativa das mudanças realizadas.

SRP – Princípio da Responsabilidade Única:

• Classe afetada: AutenticadorDeLoginDAO.
- Antes: verificava login, tratava erros, exibia mensagens e fazia lógica de decisão.
- Depois: agora somente autentica e retorna o tipo de usuário.
- Justificativa: reduz acoplamento e deixa mais fácil testar e manter o código.

```
14  public String verificarLogin(String usuario, String senha) {
15      try {
16          String sqlAdm = "select 1 from loginAdm where usuario = ? and senha = ?";
17          try (PreparedStatement psAdm = con.prepareStatement(sqlAdm)) {
18              psAdm.setString(1, usuario);
19              psAdm.setString(2, senha);
20              try (ResultSet rsAdm = psAdm.executeQuery()) {
21                  if (rsAdm.next()) {
22                      return "administrador";
23                  }
24              }
25          }
26
27          String sqlUser = "select 1 from loginUser where usuario = ? and senha = ?";
28          try (PreparedStatement psUser = con.prepareStatement(sqlUser)) {
29              psUser.setString(1, usuario);
30              psUser.setString(2, senha);
31              try (ResultSet rsUser = psUser.executeQuery()) {
32                  if (rsUser.next()) {
33                      return "leitor";
34                  }
35              }
36          }
37
38      } catch (Exception e) {
39          e.printStackTrace();
40          return "erro";
41      }
42
43      return "nao_encontrado";
44  }
```

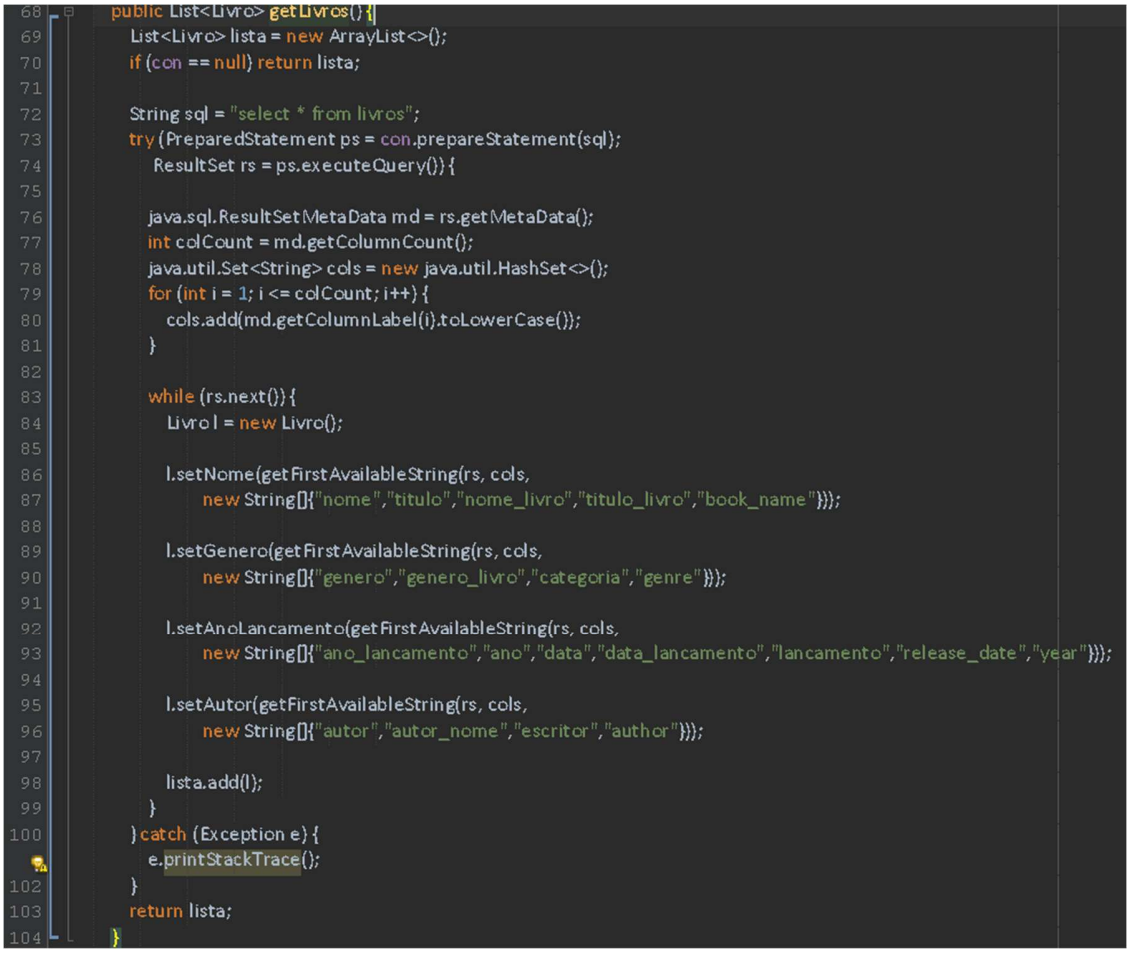
OCP – Aberto para Extensão, Fechado para Modificação:

• Classe afetada: RegistroDeLivroDAO.

- Foi criada a lógica de mapeamento dinâmico de colunas (*firstMatch*), permitindo aceitar bancos diferentes sem alterar o código principal.

- Justificativa: novas colunas podem ser adicionadas ao banco sem precisar reescrever o método *getLivros*().

```
25    public RegistroDeLivroDAO() {
26        ConexaoBdDAO conexao = new ConexaoBdDAO();
27        if (conexao.conectar()) {
28            this.con = conexao.getCon();
29            this.createdInternally = true;
30        } else {
31            this.con = null;
32            this.createdInternally = false;
33        }
34        ensureColumnMap();
35    }
36    private void ensureColumnMap() {
37        columnMap.clear();
38        if (con == null) return;
39
40        String sampleSql = "select * from livros limit 1";
41        try (PreparedStatement ps = con.prepareStatement(sampleSql);
42             ResultSet rs = ps.executeQuery()) {
43
44            java.sql.ResultSetMetaData md = rs.getMetaData();
45            int colCount = md.getColumnCount();
46            java.util.Set<String> cols = new java.util.HashSet<>();
47            for (int i = 1; i <= colCount; i++) {
48                cols.add(md.getColumnLabel(i).toLowerCase());
49            }
50
51            columnMap.put("nome", firstMatch(cols, new String[]{"nome","titulo","nome_livro","titulo_livro","book_name"}));
52            columnMap.put("genero", firstMatch(cols, new String[]{"genero","genero_livro","categoria","genre"}));
53            columnMap.put("ano_lancamento", firstMatch(cols, new String[]{"ano_lancamento","ano","data","data_lancamento","lancamento","release_date","year"}));
54            columnMap.put("autor", firstMatch(cols, new String[]{"autor","autor_nome","escritor","author"}));
55
56        } catch (Exception e) {
57            e.printStackTrace();
58        }
59    }
```

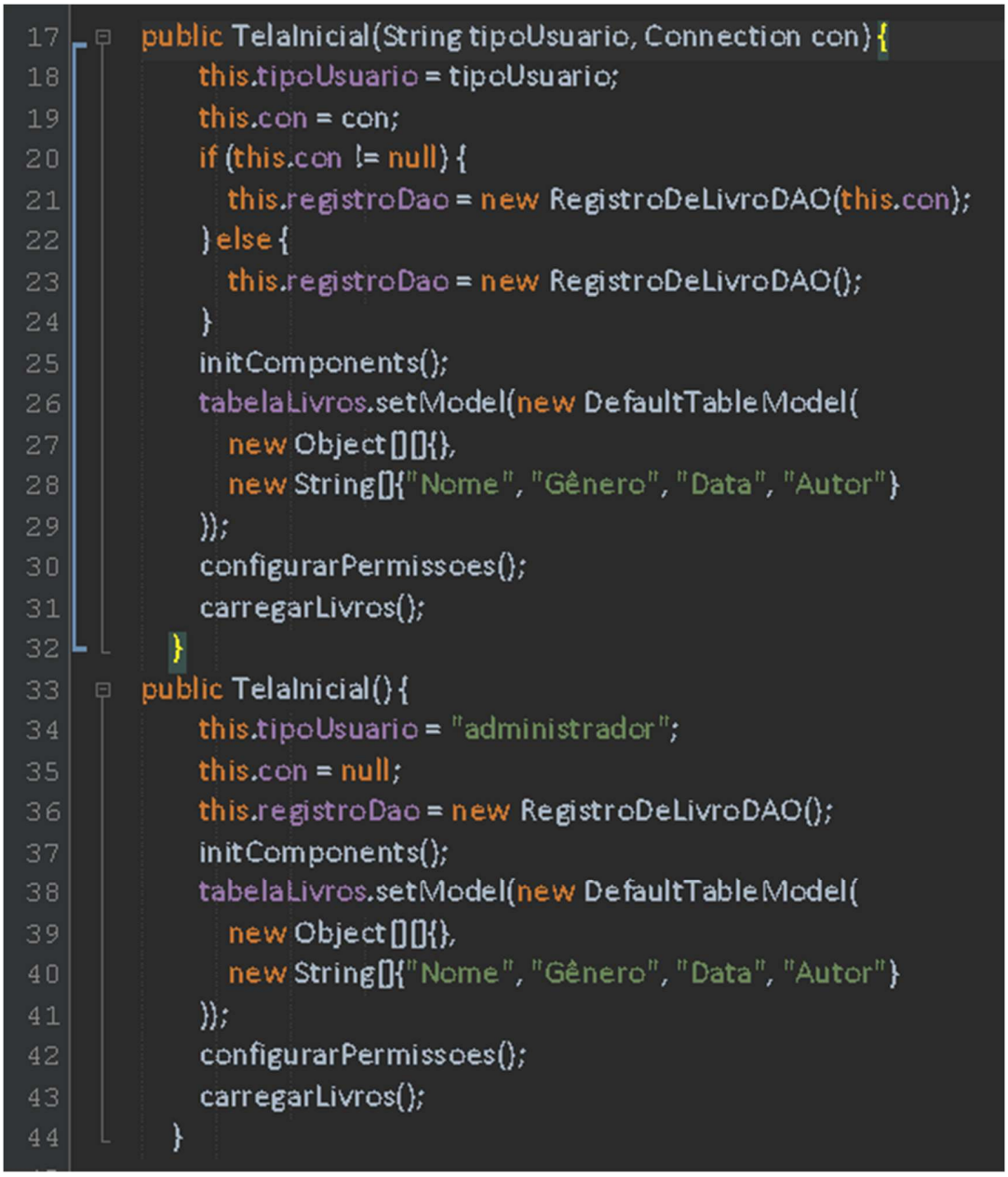


```
68    public List<Livro> getLivros(){
69        List<Livro> lista = new ArrayList<>();
70        if (con == null) return lista;
71
72        String sql = "select * from livros";
73        try (PreparedStatement ps = con.prepareStatement(sql);
74             ResultSet rs = ps.executeQuery()) {
75
76            java.sql.ResultSetMetaData md = rs.getMetaData();
77            int colCount = md.getColumnCount();
78            java.util.Set<String> cols = new java.util.HashSet<>();
79            for (int i = 1; i <= colCount; i++) {
80                cols.add(md.getColumnLabel(i).toLowerCase());
81            }
82
83            while (rs.next()) {
84                Livro l = new Livro();
85
86                l.setNome(getFirstAvailableString(rs, cols,
87                        new String[]{"nome","titulo","nome_livro","titulo_livro","book_name"}));
88
89                l.setGenero(getFirstAvailableString(rs, cols,
90                        new String[]{"genero","genero_livro","categoria","genre"}));
91
92                l.setAnoLancamento(getFirstAvailableString(rs, cols,
93                        new String[]{"ano_lancamento","ano","data","data_lancamento","lancamento","release_date","year"}));
94
95                l.setAutor(getFirstAvailableString(rs, cols,
96                        new String[]{"autor","autor_nome","escritor","author"}));
97
98                lista.add(l);
99            }
100       } catch (Exception e) {
101           e.printStackTrace();
102       }
103       return lista;
104   }
```

DIP – Princípio da Inversão de Dependência:

• Classes afetadas: Telainicial e RegistroDeLivroDAO.

- A Telainicial não cria a conexão diretamente; ela recebe o *Connection* pronto.
- Justificativa: facilita testes, simulação de banco e troca futura do provedor de dados.



```
17  public TelaInicial(String tipoUsuario, Connection con){
18      this.tipoUsuario = tipoUsuario;
19      this.con = con;
20      if (this.con != null) {
21          this.registroDao = new RegistroDeLivroDAO(this.con);
22      }else {
23          this.registroDao = new RegistroDeLivroDAO();
24      }
25      initComponents();
26      tabelaLivros.setModel(new DefaultTableModel(
27          new Object[][]{},
28          new String[]{"Nome", "Gênero", "Data", "Autor"}
29      ));
30      configurarPermissoes();
31      carregarLivros();
32    }
33  public TelaInicial(){
34      this.tipoUsuario = "administrador";
35      this.con = null;
36      this.registroDao = new RegistroDeLivroDAO();
37      initComponents();
38      tabelaLivros.setModel(new DefaultTableModel(
39          new Object[][]{},
40          new String[]{"Nome", "Gênero", "Data", "Autor"}
41      ));
42      configurarPermissoes();
43      carregarLivros();
44    }
```

Refatorações Aplicadas:

Separação de Responsabilidades:

• AutenticadorDeLoginDAO passou por limpeza de código, remoção de lógica visual e concentração apenas no processo de autenticação.

```java
private void ensureColumnMap() {
    columnMap.clear();
    if (con == null) return;

    String sampleSql = "select * from livros limit 1";
    try (PreparedStatement ps = con.prepareStatement(sampleSql);
         ResultSet rs = ps.executeQuery()) {

        java.sql.ResultSetMetaData md = rs.getMetaData();
        int colCount = md.getColumnCount();
        java.util.Set<String> cols = new java.util.HashSet<>();
        for (int i = 1; i <= colCount; i++) {
            cols.add(md.getColumnLabel(i).toLowerCase());
        }

        columnMap.put("nome", firstMatch(cols, new String[]{"nome","titulo","nome_livro","titulo_livro","book_name"});
        columnMap.put("genero", firstMatch(cols, new String[]{"genero","genero_livro","categoria","genre"});
        columnMap.put("ano_lancamento", firstMatch(cols, new String[]{"ano_lancamento","ano","data","data_lancamento","lancamento","release_date","year"});
        columnMap.put("autor", firstMatch(cols, new String[]{"autor","autor_nome","escritor","author"});

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Padronização do RegistroDeLivroDAO:

• Refatoração do método *getLivros*() para evitar quebras caso o nome das colunas varie.
• Criação da função *getFirstAvailableString*().
• Justificativa: melhora compatibilidade e reduz repetição de código.

```java
private String getFirstAvailableString(ResultSet rs, java.util.Set<String> cols, String[] options) {
    try {
        for (String opt : options) {
            if (cols.contains(opt.toLowerCase())) {
                String v = rs.getString(opt);
                return v != null ? v : "";
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

Classe Telainicial reorganizada:

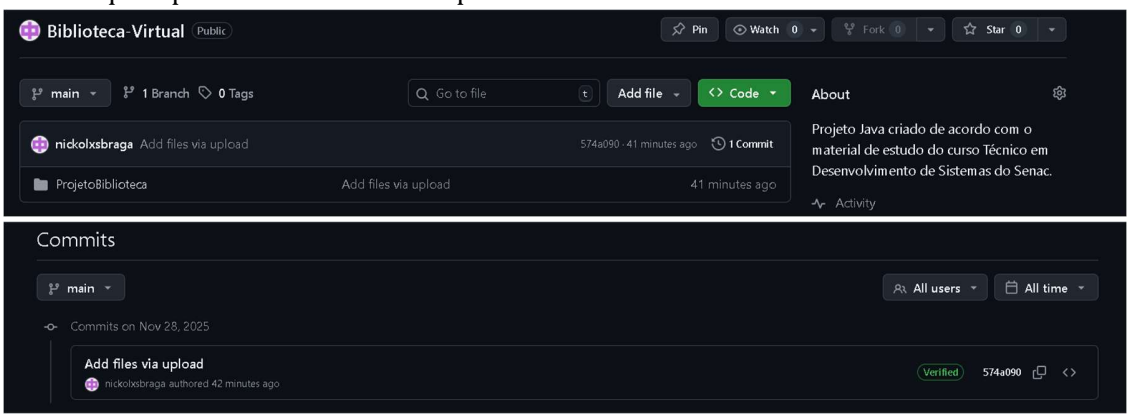
• A lógica de permissões (configurarPermissoes) foi isolada do construtor.
• Isso melhora organização, clareza e mantém o SRP.

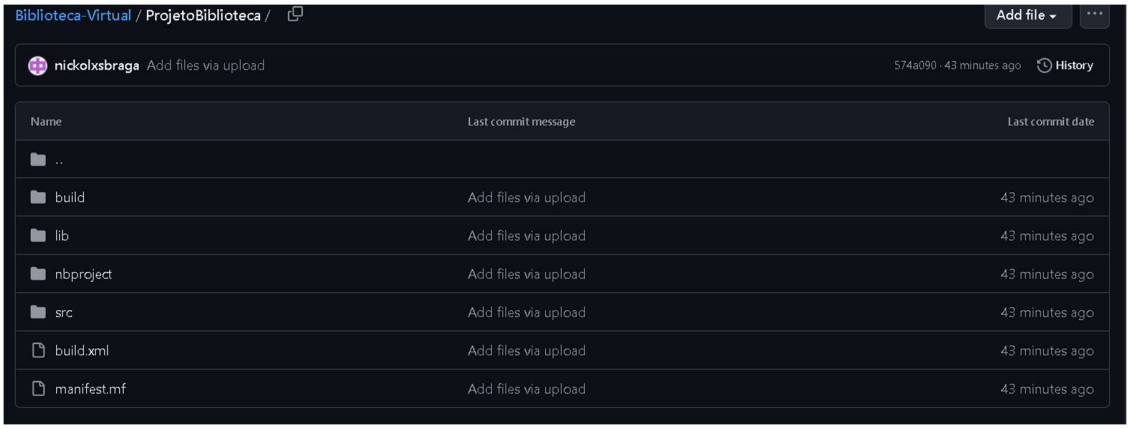
```java
private void configurarPermissoes() {
    if ("leitor".equals(tipoUsuario)) {
        botaoAdicionarLivro.setEnabled(false);
        botaoRemoverLivro.setEnabled(false);
    } else {
        botaoAdicionarLivro.setEnabled(true);
        botaoRemoverLivro.setEnabled(true);
    }
}
```

```java
public TelaInicial(String tipoUsuario, Connection con) {
    this.tipoUsuario = tipoUsuario;
    this.con = con;
    if (this.con != null) {
        this.registroDao = new RegistroDeLivroDAO(this.con);
    } else {
        this.registroDao = new RegistroDeLivroDAO();
    }
    initComponents();
    tabelaLivros.setModel(new DefaultTableModel(
        new Object[][]{},
        new String[]{"Nome", "Gênero", "Data", "Autor"}
    ));
    configurarPermissoes();
    carregarLivros();
}
```

Evidências do Repositório *GitHub*:

Inclua aqui capturas de tela do seu repositório *GitHub* contendo:




Biblioteca-Virtual / ProjetoBiblioteca /
Add file
nickolxsbraga Add files via upload
574a090 · 43 minutes ago
History
Name
Last commit message
Last commit date
..
build
Add files via upload
43 minutes ago
lib
Add files via upload
43 minutes ago
nbproject
Add files via upload
43 minutes ago
src
Add files via upload
43 minutes ago
build.xml
Add files via upload
43 minutes ago
manifest.mf
Add files via upload
43 minutes ago