

# Sistema de Gestão de Biblioteca

## Sumário

<b>Sumário.....</b>	<b>1</b>
<b>Resumo.....</b>	<b>1</b>
<b>1. Introdução.....</b>	<b>2</b>
<b>2. Metodologia.....</b>	<b>2</b>
<b>3. Estrutura do Banco de Dados.....</b>	<b>2</b>
3.1. Tabelas e relacionamento.....	2
3.2 Diagrama do banco de dados.....	3
<b>4. Páginas em JSP e Classes Java.....</b>	<b>3</b>
4.1 Páginas criadas.....	3
4.1.1 Páginas do tipo form.....	3
4.1.2 Páginas do tipo code.....	4
4.1.3 Páginas do tipo consulta.....	6
4.2 Classes JAVA.....	7
4.2.1 Estrutura de pacotes.....	7
4.2.2 Classe para acesso ao banco de dados.....	7
4.2.3 Estrutura das classes do tipo Data Access Object.....	8
<b>5. Testes e Validação.....</b>	<b>8</b>
<b>6. Resultados e Conclusão.....</b>	<b>8</b>
<b>7. Trabalhos Futuros.....</b>	<b>9</b>

## Resumo

Este trabalho tem como objetivo a entrega de um projeto de sistema para a gestão de uma biblioteca de ensino, desenvolvido com o propósito de automatizar e facilitar as atividades rotineiras de uma biblioteca. O sistema tem como objetivo principal organizar e controlar o acervo de livros, gerenciar empréstimos, facilitar a busca de obras e fornecer relatórios relevantes. O projeto envolveu a modelagem de processos da biblioteca, a criação de um diagrama de banco de dados, um diagrama de classes, a implementação da estrutura do banco de dados e regras de negócio do mesmo, o desenvolvimento das páginas em JSP e a criação das classes Java necessárias para o funcionamento do sistema.

# 1. Introdução

As bibliotecas desempenham um papel fundamental na disseminação do conhecimento, e é essencial que elas possuam um sistema eficiente para gerenciar suas atividades diárias. Este trabalho propõe um sistema de gestão de biblioteca que visa automatizar e otimizar processos como cadastro de livros, controle de empréstimos, busca de obras e geração de relatórios. O sistema busca melhorar a experiência dos usuários da biblioteca, tornando mais fácil e ágil o acesso aos recursos disponíveis.

## 2. Metodologia

O diagrama de processos foi criado visando estruturar e levantar os requisitos necessários para o sistema. O diagrama de banco de dados foi elaborado para modelar a estrutura do sistema, definindo as tabelas, os relacionamentos e as chaves estrangeiras necessárias. O diagrama de classes foi criado para representar a estrutura das classes do sistema, definindo os atributos e métodos de cada classe.

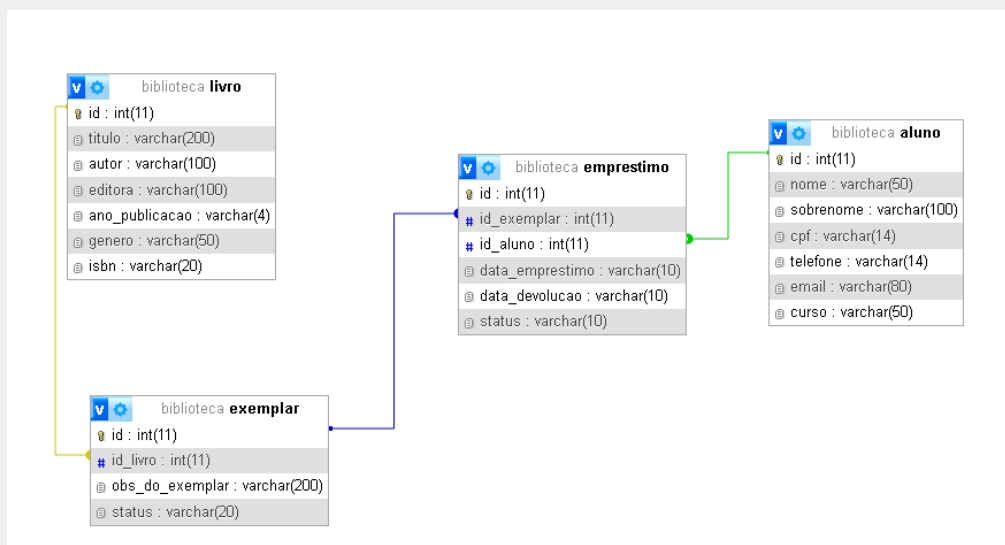
## 3. Estrutura do Banco de Dados

O banco de dados foi implementado utilizando o Sistema de Gerenciamento de Banco de Dados Relacional (SGBD) MySQL. Foram criadas tabelas para armazenar informações sobre livros, exemplares, alunos e empréstimos. Chaves primárias e chaves estrangeiras foram estabelecidas para garantir a integridade dos dados. Consultas SQL foram utilizadas para realizar operações de busca, inserção, atualização e exclusão no banco de dados. Durante a implementação, a Inteligência Artificial ChatGPT auxiliou no tratamento de erros de JSP e Java, contribuindo para a eficiência e qualidade do sistema.

### 3.1. Tabelas e relacionamento

Foram criadas 4 tabelas de banco de dados que se relacionam em alguma medida, sendo elas: aluno, livro, exemplar e empréstimo.

## 3.2 Diagrama do banco de dados



## 4. Páginas em JSP e Classes Java

As páginas em JSP foram desenvolvidas para fornecer uma interface amigável e intuitiva aos usuários do sistema, aproveitando e usufruindo dos recursos fornecidos pelo framework Bootstrap. Foram criadas páginas para realizar cadastro e consulta de livros por título e autor; cadastro e consulta de exemplares por livro, disponibilidade e observações adicionais; registro e consulta de empréstimos por data de empréstimo e reserva, bem como por status; registro e consulta de alunos por nome e CPF. As páginas em JSP utilizam as classes Java correspondentes para processar as informações inseridas pelos usuários, realizar operações no banco de dados e retornar resultados.

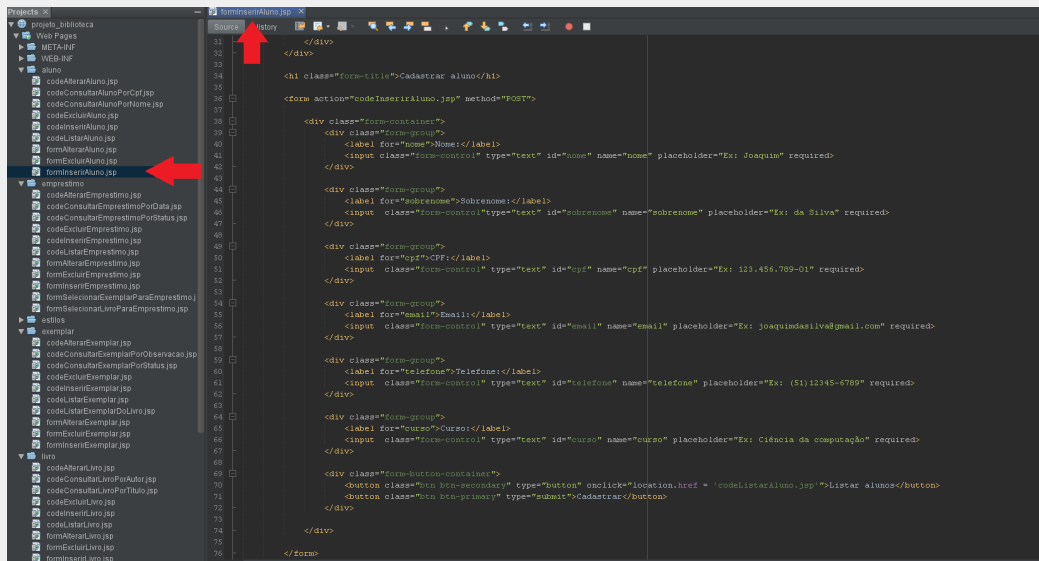
### 4.1 Páginas criadas

Foram criadas páginas com a finalidade de realizar as 4 operações básicas de um sistema de banco de dados (CRUD) com todas as tabelas incluídas.

#### 4.1.1 Páginas do tipo form

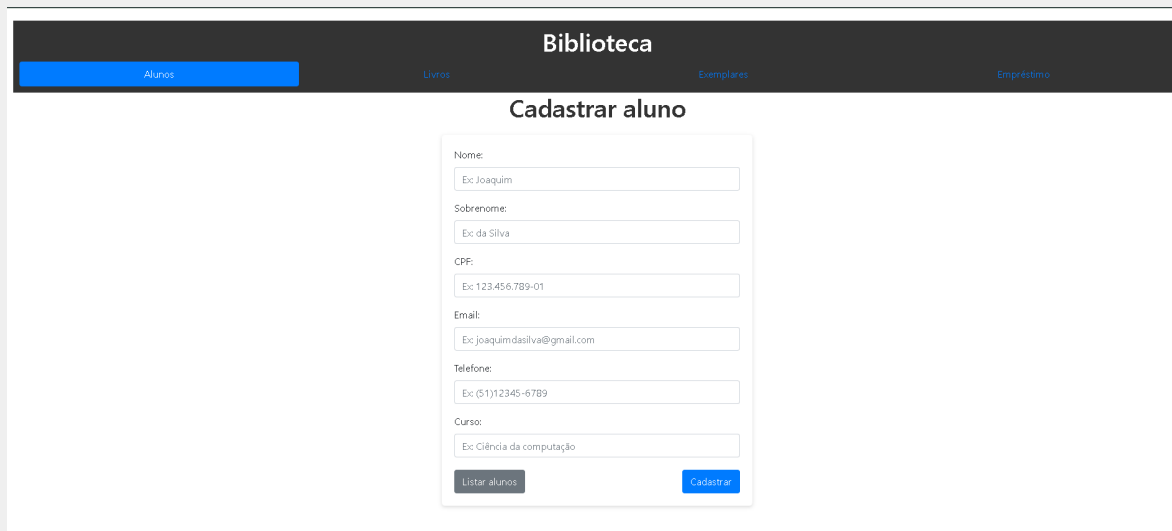
Foram criadas páginas de formulário (onde recebem input do usuário para cadastro, alteração ou exclusão de dados), estas páginas possuem o prefixo “form” no nome.

## CÓDIGO:



```
31 </div>
32 </div>
33
34 <h1 class="form-title">Cadastrar aluno</h1>
35
36 <form action="codeInsereAluno.jsp" method="POST">
37
38   <div class="form-group">
39     <div class="form-group">
40       <label for="nome">Nome:</label>
41       <input class="form-control" type="text" id="nome" name="nome" placeholder="Ex: Joaquim" required>
42     </div>
43   </div>
44
45   <div class="form-group">
46     <label for="sobrenome">Sobrenome:</label>
47     <input class="form-control" type="text" id="sobrenome" name="sobrenome" placeholder="Ex: da Silva" required>
48   </div>
49
50   <div class="form-group">
51     <label for="cpf">CPF:</label>
52     <input class="form-control" type="text" id="cpf" name="cpf" placeholder="Ex: 123.456.789-01" required>
53   </div>
54
55   <div class="form-group">
56     <label for="email">Email:</label>
57     <input class="form-control" type="text" id="email" name="email" placeholder="Ex: joaquindasilva@gmail.com" required>
58   </div>
59
60   <div class="form-group">
61     <label for="telefone">Telefone:</label>
62     <input class="form-control" type="text" id="telefone" name="telefone" placeholder="Ex: (51)12345-6789" required>
63   </div>
64
65   <div class="form-group">
66     <label for="curso">Curso:</label>
67     <input class="form-control" type="text" id="curso" name="curso" placeholder="Ex: Ciência da computação" required>
68   </div>
69
70   <div class="form-button-container">
71     <button class="btn btn-secondary" type="button" onclick="location.href = 'codeListarAluno.jsp'">Listar alunos</button>
72     <button class="btn btn-primary" type="submit">Cadastrar</button>
73   </div>
74 </div>
75 </form>
76 </div>
```

## LAYOUT:



**Biblioteca**

[Alunos](#) [Livros](#) [Exemplares](#) [Empréstimo](#)

**Cadastrar aluno**

Nome:

Sobrenome:

CPF:

Email:

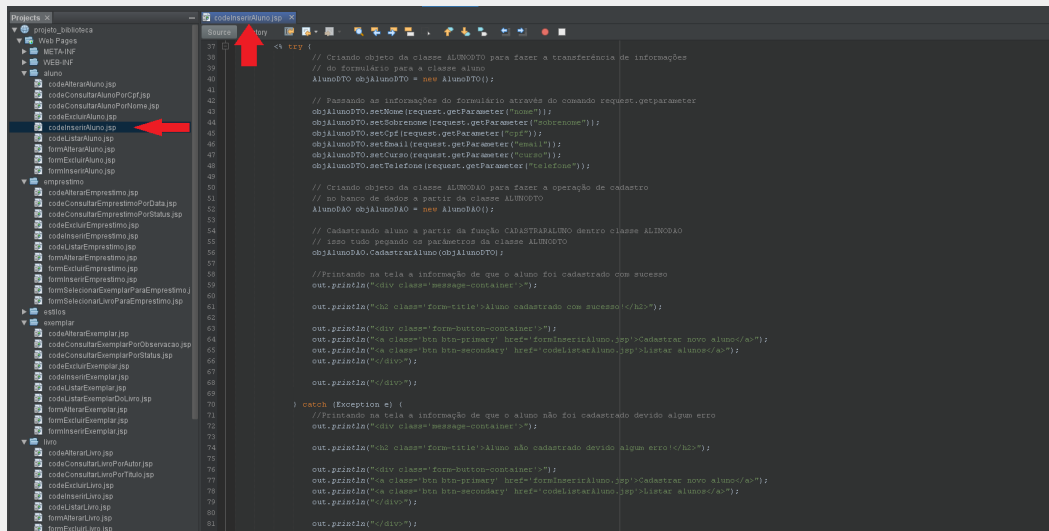
Telefone:

Curso:

### 4.1.2 Páginas do tipo code

As páginas do tipo form realizam um direcionamento para páginas de codificação (onde são feitos os tratamentos com as informações passadas via formulário), essas páginas possuem um prefixo “code” no nome. Estas páginas trabalham com as classes Java para realizar operações onde o banco de dados é necessário.

## CÓDIGO JSP:



```

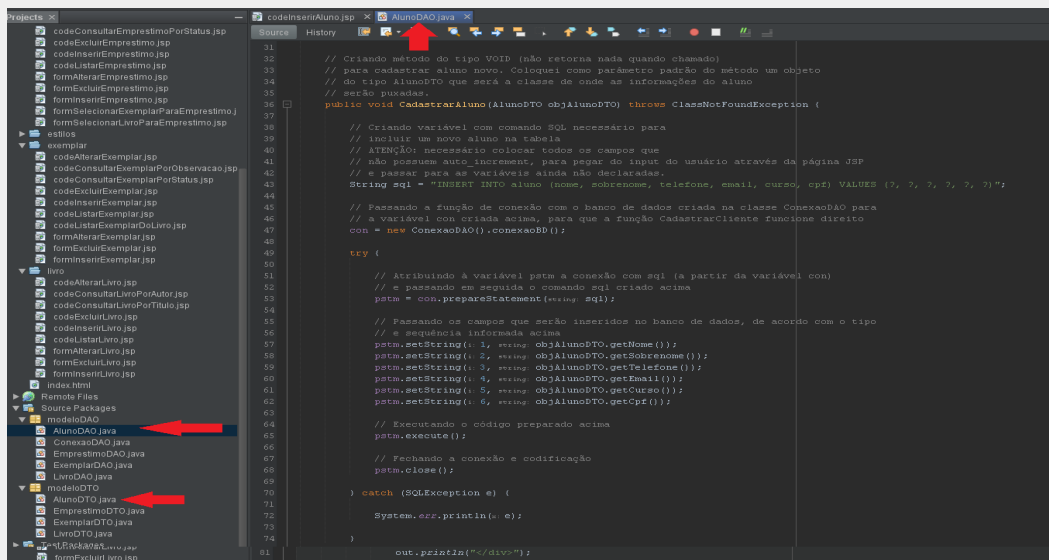
37 try {
38     // Criando objeto da classe ALUNOTO para fazer a transferência de informações
39     // do formulário para a classe aluno
40     AlunoDTO objAlunoTO = new AlunoTO();
41
42     // Passando as informações do formulário através do comando request.getParameter
43     objAlunoTO.setNome(request.getParameter("nome"));
44     objAlunoTO.setSobrenome(request.getParameter("sobrenome"));
45     objAlunoTO.setCpf(request.getParameter("cpf"));
46     objAlunoTO.setEmail(request.getParameter("email"));
47     objAlunoTO.setCurso(request.getParameter("curso"));
48     objAlunoTO.setTelefone(request.getParameter("telefone"));
49
50     // Criando objeto da classe ALUNODAO para fazer a operação de cadastro
51     // no banco de dados a partir da classe ALUNOTO
52     AlunoDAO objAlunoDAO = new AlunoDAO();
53
54     // Cadastro de aluno a partir da função CADASTRARALUNO dentro classe ALUNODAO
55     objAlunoDAO.CadastrearAluno(objAlunoTO);
56
57     //Printando na tela a informação de que o aluno foi cadastrado com sucesso
58     out.println("div classe= message-container");
59
60     out.println("<div classe= form-title>Aluno cadastrado com sucesso</div>");
61
62     out.println("<div classe= form-button-container>");
63     out.println("<a classe= btn btn-primary href= formInserirAluno.jsp>Cadastrar novo aluno</a>");
64     out.println("<a classe= btn btn-secondary href= codListarAluno.jsp>Listar alunos</a>");
65     out.println("</div>");
66
67     } catch (Exception e) {
68         //Printando na tela a informação de que o aluno não foi cadastrado devido algum erro
69         out.println("div classe= message-container");
70
71         out.println("<div classe= form-title>Aluno não cadastrado devido algum erro</div>");
72
73         out.println("<div classe= form-button-container>");
74         out.println("<a classe= btn btn-primary href= formInserirAluno.jsp>Cadastrar novo aluno</a>");
75         out.println("<a classe= btn btn-secondary href= codListarAluno.jsp>Listar alunos</a>");
76         out.println("</div>");
77
78     }
79
80     out.println("</div>");
81 }

```

## LAYOUT:



## CLASSE JAVA:



```

31 // Criando método do tipo VOID (não retorna nada quando chamado)
32 // para cadastrar aluno novo. Coloquei como parametro padrão do método um objeto
33 // do tipo AlunoDTO que será a classe de onde as informações do aluno
34 // serão puxadas.
35 public void CadastrearAluno(AlunoDTO objAlunoTO) throws ClassNotFoundException {
36
37     // Criando variável com comando SQL necessário para
38     // incluir um novo aluno na tabela
39     // ATENÇÃO: necessário colocar todos os campos que
40     // não possuem auto_increment, para pegar do input do usuário através da página JSP
41     // e passar para as variáveis ainda não declaradas.
42     String sql = "INSERT INTO aluno (nome, sobrenome, telefone, email, curso, cpf) VALUES (?, ?, ?, ?, ?, ?)";
43
44     // Passando a função de conexão com o banco de dados criada na classe ConexaoDAO para
45     // a variável con criada acima, para que a função CadastrearAluno funcione direito
46     con = new ConexaoDAO().conectarBD();
47
48     try {
49
50         // Atribuindo à variável pstmt a conexão com sql (a partir da variável con)
51         // e passando em seguida o comando sql criado acima
52         pstmt = con.prepareStatement(sql);
53
54         // Passando os campos que serão inseridos no banco de dados, de acordo com o tipo
55         // e sequência informada acima
56         pstmt.setString(1, objAlunoTO.getNome());
57         pstmt.setString(2, objAlunoTO.getSobrenome());
58         pstmt.setString(3, objAlunoTO.getTelefone());
59         pstmt.setString(4, objAlunoTO.getEmail());
60         pstmt.setString(5, objAlunoTO.getCurso());
61         pstmt.setString(6, objAlunoTO.getCpf());
62
63         // Executando o código preparado acima
64         pstmt.executeUpdate();
65
66         // Fechando a conexão e codificação
67         pstmt.close();
68
69     } catch (SQLException e) {
70         System.err.println(e);
71     }
72
73     out.println("</div>");
74 }

```

As páginas do tipo consulta servem para listar informações trazidas do banco de dados e recebem o mesmo prefixo “code” na nomenclatura.

```

49  <div class="form-group">
50      <input type="text" value="" class="form-control" />
51      <input type="button" value="Pesquisar" class="btn btn-primary" />
52      <input type="button" value="Limpar" class="btn btn-danger" />
53  </div>
54  <div class="table">
55      <table>
56          <thead>
57              <tr>
58                  <th>Nome</th>
59                  <th>CPF</th>
60                  <th>Telefone</th>
61                  <th>Email</th>
62                  <th>Cidade</th>
63              </tr>
64          </thead>
65          <tbody>
66              <tr>
67                  <td>{{ $nome }}</td>
68                  <td>{{ $cpf }}</td>
69                  <td>{{ $telefone }}</td>
70                  <td>{{ $email }}</td>
71                  <td>{{ $cidade }}</td>
72              </tr>
73          </tbody>
74      </table>
75  </div>
76  <div class="form-group">
77      <input type="text" value="" class="form-control" />
78      <input type="button" value="Pesquisar" class="btn btn-primary" />
79      <input type="button" value="Limpar" class="btn btn-danger" />
80  </div>
81  <div class="table">
82      <table>
83          <thead>
84              <tr>
85                  <th>Nome</th>
86                  <th>CPF</th>
87                  <th>Telefone</th>
88                  <th>Email</th>
89                  <th>Cidade</th>
90              </tr>
91          </thead>
92          <tbody>
93              <tr>
94                  <td>{{ $nome }}</td>
95                  <td>{{ $cpf }}</td>
96                  <td>{{ $telefone }}</td>
97                  <td>{{ $email }}</td>
98                  <td>{{ $cidade }}</td>
99              </tr>
100          </tbody>
101      </table>
102  </div>
103  <div class="form-group">
104      <input type="text" value="" class="form-control" />
105      <input type="button" value="Pesquisar" class="btn btn-primary" />
106      <input type="button" value="Limpar" class="btn btn-danger" />
107  </div>
108  <div class="table">
109      <table>
110          <thead>
111              <tr>
112                  <th>Nome</th>
113                  <th>CPF</th>
114                  <th>Telefone</th>
115                  <th>Email</th>
116                  <th>Cidade</th>
117              </tr>
118          </thead>
119          <tbody>
120              <tr>
121                  <td>{{ $nome }}</td>
122                  <td>{{ $cpf }}</td>
123                  <td>{{ $telefone }}</td>
124                  <td>{{ $email }}</td>
125                  <td>{{ $cidade }}</td>
126              </tr>
127          </tbody>
128      </table>
129  </div>
130  <div class="form-group">
131      <input type="text" value="" class="form-control" />
132      <input type="button" value="Pesquisar" class="btn btn-primary" />
133      <input type="button" value="Limpar" class="btn btn-danger" />
134  </div>
135  <div class="table">
136      <table>
137          <thead>
138              <tr>
139                  <th>Nome</th>
140                  <th>CPF</th>
141                  <th>Telefone</th>
142                  <th>Email</th>
143                  <th>Cidade</th>
144              </tr>
145          </thead>
146          <tbody>
147              <tr>
148                  <td>{{ $nome }}</td>
149                  <td>{{ $cpf }}</td>
150                  <td>{{ $telefone }}</td>
151                  <td>{{ $email }}</td>
152                  <td>{{ $cidade }}</td>
153              </tr>
154          </tbody>
155      </table>
156  </div>
157  <div class="form-group">
158      <input type="text" value="" class="form-control" />
159      <input type="button" value="Pesquisar" class="btn btn-primary" />
160      <input type="button" value="Limpar" class="btn btn-danger" />
161  </div>
162  <div class="table">
163      <table>
164          <thead>
165              <tr>
166                  <th>Nome</th>
167                  <th>CPF</th>
168                  <th>Telefone</th>
169                  <th>Email</th>
170                  <th>Cidade</th>
171              </tr>
172          </thead>
173          <tbody>
174              <tr>
175                  <td>{{ $nome }}</td>
176                  <td>{{ $cpf }}</td>
177                  <td>{{ $telefone }}</td>
178                  <td>{{ $email }}</td>
179                  <td>{{ $cidade }}</td>
180              </tr>
181          </tbody>
182      </table>
183  </div>
184  <div class="form-group">
185      <input type="text" value="" class="form-control" />
186      <input type="button" value="Pesquisar" class="btn btn-primary" />
187      <input type="button" value="Limpar" class="btn btn-danger" />
188  </div>
189  <div class="table">
190      <table>
191          <thead>
192              <tr>
193                  <th>Nome</th>
194                  <th>CPF</th>
195                  <th>Telefone</th>
196                  <th>Email</th>
197                  <th>Cidade</th>
198              </tr>
199          </thead>
200          <tbody>
201              <tr>
202                  <td>{{ $nome }}</td>
203                  <td>{{ $cpf }}</td>
204                  <td>{{ $telefone }}</td>
205                  <td>{{ $email }}</td>
206                  <td>{{ $cidade }}</td>
207              </tr>
208          </tbody>
209      </table>
210  </div>
211  <div class="form-group">
212      <input type="text" value="" class="form-control" />
213      <input type="button" value="Pesquisar" class="btn btn-primary" />
214      <input type="button" value="Limpar" class="btn btn-danger" />
215  </div>
216  <div class="table">
217      <table>
218          <thead>
219              <tr>
220                  <th>Nome</th>
221                  <th>CPF</th>
222                  <th>Telefone</th>
223                  <th>Email</th>
224                  <th>Cidade</th>
225              </tr>
226          </thead>
227          <tbody>
228              <tr>
229                  <td>{{ $nome }}</td>
230                  <td>{{ $cpf }}</td>
231                  <td>{{ $telefone }}</td>
232                  <td>{{ $email }}</td>
233                  <td>{{ $cidade }}</td>
234              </tr>
235          </tbody>
236      </table>
237  </div>
238  <div class="form-group">
239      <input type="text" value="" class="form-control" />
240      <input type="button" value="Pesquisar" class="btn btn-primary" />
241      <input type="button" value="Limpar" class="btn btn-danger" />
242  </div>
243  <div class="table">
244      <table>
245          <thead>
246              <tr>
247                  <th>Nome</th>
248                  <th>CPF</th>
249                  <th>Telefone</th>
250                  <th>Email</th>
251                  <th>Cidade</th>
252              </tr>
253          </thead>
254          <tbody>
255              <tr>
256                  <td>{{ $nome }}</td>
257                  <td>{{ $cpf }}</td>
258                  <td>{{ $telefone }}</td>
259                  <td>{{ $email }}</td>
260                  <td>{{ $cidade }}</td>
261              </tr>
262          </tbody>
263      </table>
264  </div>
265  <div class="form-group">
266      <input type="text" value="" class="form-control" />
267      <input type="button" value="Pesquisar" class="btn btn-primary" />
268      <input type="button" value="Limpar" class="btn btn-danger" />
269  </div>
270  <div class="table">
271      <table>
272          <thead>
273              <tr>
274                  <th>Nome</th>
275                  <th>CPF</th>
276                  <th>Telefone</th>
277                  <th>Email</th>
278                  <th>Cidade</th>
279              </tr>
280          </thead>
281          <tbody>
282              <tr>
283                  <td>{{ $nome }}</td>
284                  <td>{{ $cpf }}</td>
285                  <td>{{ $telefone }}</td>
286                  <td>{{ $email }}</td>
287                  <td>{{ $cidade }}</td>
288              </tr>
289          </tbody>
290      </table>
291  </div>
292  <div class="form-group">
293      <input type="text" value="" class="form-control" />
294      <input type="button" value="Pesquisar" class="btn btn-primary" />
295      <input type="button" value="Limpar" class="btn btn-danger" />
296  </div>
297  <div class="table">
298      <table>
299          <thead>
300              <tr>
301                  <th>Nome</th>
302                  <th>CPF</th>
303                  <th>Telefone</th>
304                  &
```

Alunos

Livros

Exemplares

Empréstimo

Biblioteca

Resultado da consulta pelo nome: Marlon

Digite o CPF...

Pesquisar por CPF

Digite o nome...

Pesquisar por nome

Código	Nome	Sobrenome	CPF	Telefone	Email	Curso	Empréstimo	Cadastro do Aluno
6	Marlon	prado	ihih	loulhol	ouhoh	hgh	<div>Insere</div> <div>Listar</div>	<div>Alterar</div> <div>Excluir</div>
7	Marlon	prado	ihih	loulhol	ouhoh	hgh	<div>Insere</div> <div>Listar</div>	<div>Alterar</div> <div>Excluir</div>

Voltar para lista completa

Insere novo aluno

The screenshot shows the code for `AlunoDAO.java` in an IDE. The code implements the `AlunoDAO` interface, including methods for listing, searching, and creating students. A red arrow points to the `AlunoDAO.java` file in the Source Packages view.

```

253 }
254 codeExcluirEmprestimo.jsp
255 codeListarEmprestimo.jsp
256 formAlterarEmprestimo.jsp
257 formExcluirEmprestimo.jsp
258 formInserirEmprestimo.jsp
259 // Metodo para consultar aluno por nome
260 public ArrayList<AlunoDTO> consultarAlunoPorNome(String nome) throws ClassNotFoundException {
261     String sql = "SELECT * FROM aluno WHERE nome LIKE ?";
262     con = new ConexaoDAO().conexaoBD();
263     try {
264         pstmt = con.prepareStatement(sql);
265         pstmt.setString(1, nome + "%");
266         rs = pstmt.executeQuery();
267         while (rs.next()) {
268             AlunoDTO objAlunoDTO = new AlunoDTO();
269             objAlunoDTO.setId(rs.getInt("id"));
270             objAlunoDTO.setNome(rs.getString("nome"));
271             objAlunoDTO.setSobrenome(rs.getString("sobrenome"));
272             objAlunoDTO.setTelefone(rs.getString("telefone"));
273             objAlunoDTO.setEmail(rs.getString("email"));
274             objAlunoDTO.setCidade(rs.getString("cidade"));
275             objAlunoDTO.setCpf(rs.getString("cpf"));
276             lista.add(objAlunoDTO);
277         }
278         pstmt.close();
279     } catch (SQLException e) {
280         System.out.println(e);
281     }
282     return lista;
283 }
284 // Criando metodo para retornar o titulo do livro a partir do ID informado
285 public String obterNomeDoLivro(int idLivro) throws ClassNotFoundException {
286     // Criando variavel para armazenar o titulo do livro
287     String nome = "";
288     // Criando comando SQL para extrair o titulo do livro a partir do ID informado
289     String sql = "SELECT nome FROM livro WHERE id = ?";

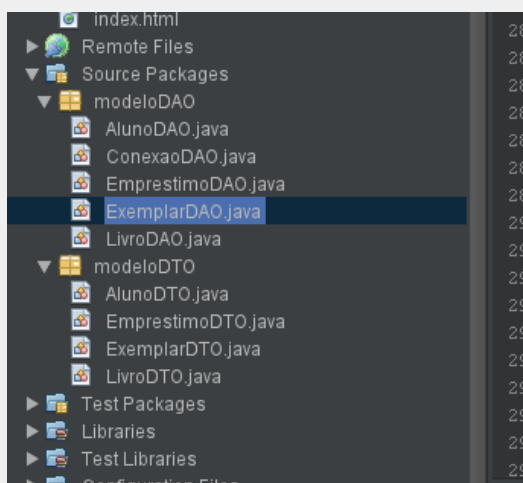
```

## 4.2 Classes JAVA

Foram criadas as classes necessárias para cada uma das tabelas da base de dados.

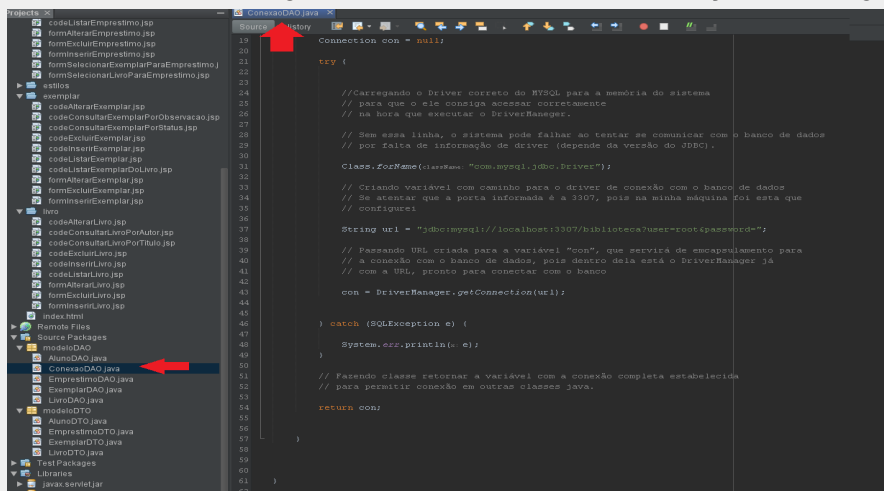
### 4.2.1 Estrutura de pacotes

As classes foram separadas em dois pacotes, o primeiro foi destinado ao tratamento de variáveis do banco de dados (Data Transfer Object, DTO) , a fim de evitar comunicação direta com as variáveis de cada classe (DTO); o segundo pacote foi destinado ao acesso do banco de dados (Data Access Object, DAO).



### 4.2.2 Classe para acesso ao banco de dados

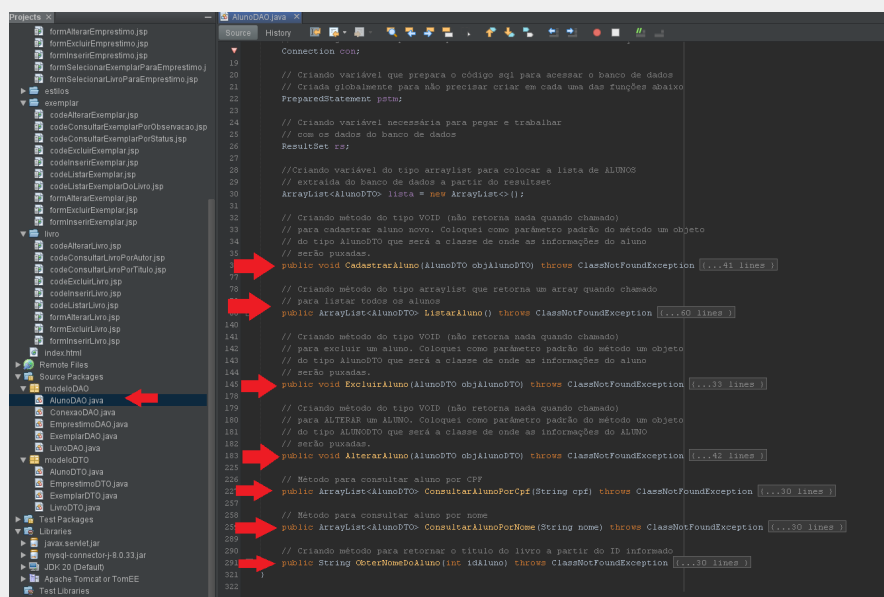
Foi criada uma classe específica para a conexão do banco de dados, com a finalidade de poupar linhas de código e otimizar a leitura e manutenção do código.



### 4.2.3 Estrutura das classes do tipo Data Access Object

Dentro de cada classe do tipo DAO existem todos os métodos necessários para trabalhar com as informações do banco de dados.

Os métodos estão separados de acordo com o banco de dados principal que ele opera, por exemplo, “consultarTituloDoLivro” pode ser usada dentro estrutura de pastas do exemplar, mas se refere ao livro, portanto o código estará dentro da classe LivroDAO.



## 5. Testes e Validação

Foram realizados testes durante o desenvolvimento do sistema para garantir que as funcionalidades estejam de acordo com os requisitos. Testes unitários foram aplicados nas classes Java para verificar o correto funcionamento dos métodos e a integração com o banco de dados. Além disso, foram conduzidos testes de usabilidade para garantir uma boa experiência do usuário.

## 6. Resultados e Conclusão

O projeto de desenvolvimento do sistema de gestão de biblioteca proporcionou uma experiência enriquecedora no aprendizado e aplicação de conceitos fundamentais no desenvolvimento de software. Durante o processo, foram realizadas etapas importantes, como o levantamento de requisitos, a criação de diagramas de banco de dados e de classes, a implementação da estrutura do banco de dados, o desenvolvimento das



páginas em JSP e a criação das classes Java necessárias para o trabalho com o banco de dados MySQL.

A elaboração do diagrama de banco de dados permitiu a modelagem eficiente da estrutura do sistema, identificando as entidades e seus relacionamentos, bem como as chaves primárias e estrangeiras necessárias para garantir a integridade dos dados.

O diagrama de classes, por sua vez, proporcionou uma representação visual da estrutura do sistema, definindo os atributos e métodos das classes envolvidas.

O desenvolvimento das páginas em JSP, aliado às classes Java correspondentes, permitiu a criação de uma interface amigável e interativa para os usuários do sistema. Através das páginas em JSP, foi possível realizar operações como cadastro de livros, consulta de exemplares e registro de empréstimos, garantindo um aprendizado sólido no uso de JSP.

Durante o processo de desenvolvimento, foram aplicados testes e validações para assegurar o correto funcionamento do sistema, bem como a usabilidade do mesmo. Testes unitários foram realizados nas classes Java, verificando a execução correta dos métodos e a integração adequada com o banco de dados. Além disso, foram conduzidos testes de usabilidade, buscando identificar possíveis melhorias na interface e garantir uma boa experiência do usuário.

No geral, o projeto do sistema de gestão de biblioteca proporcionou um aprendizado valioso sobre a criação de um software completo, desde a concepção inicial até a implementação final. Durante o desenvolvimento, foram aplicados conceitos importantes como orientação a objetos, levantamento de requisitos, criação de diagramas, uso de tecnologias como JSP e Java, e realização de testes de software. Essas habilidades adquiridas são fundamentais para o desenvolvimento de sistemas robustos e eficientes em diferentes contextos. Portanto, este projeto de software de gestão de biblioteca não apenas proporcionou uma solução prática para as atividades de uma biblioteca, mas também permitiu o aprimoramento dos conhecimentos e habilidades técnicas necessárias para a criação de softwares de qualidade em geral.

O desenvolvimento do sistema demonstrou a importância de uma abordagem metodológica, a utilização adequada de tecnologias e a aplicação de boas práticas de desenvolvimento de software. Esses aprendizados serão úteis em projetos futuros, contribuindo para a construção de soluções eficientes e inovadoras.

## 7. Trabalhos Futuros

Como trabalhos futuros, sugere-se a implementação de funcionalidades adicionais, como a disponibilização do sistema em dispositivos móveis e a criação de reservas.

Em suma, o sistema de gestão de biblioteca apresentado neste trabalho é uma solução abrangente e eficiente para melhorar o gerenciamento de bibliotecas, oferecendo recursos de organização, controle e acesso rápido a informações relevantes. A integração da Inteligência Artificial ChatGPT no tratamento de erros de JSP e Java contribuiu para a qualidade e eficiência do sistema, proporcionando uma experiência de desenvolvimento mais eficaz e livre de erros.