

Sistema de Gestão de Biblioteca

Sumário

Sumário.....	1
1. Introdução.....	1
2. Metodologia.....	2
3. Estrutura do Banco de Dados.....	2
3.1. Tabelas e relacionamento.....	2
3.2 Diagrama do banco de dados.....	3
4. Páginas em JSP e Classes Java.....	3
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
5. Testes e Validação.....	8
6. Resultados e Conclusão.....	8
7. Trabalhos Futuros.....	9

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.

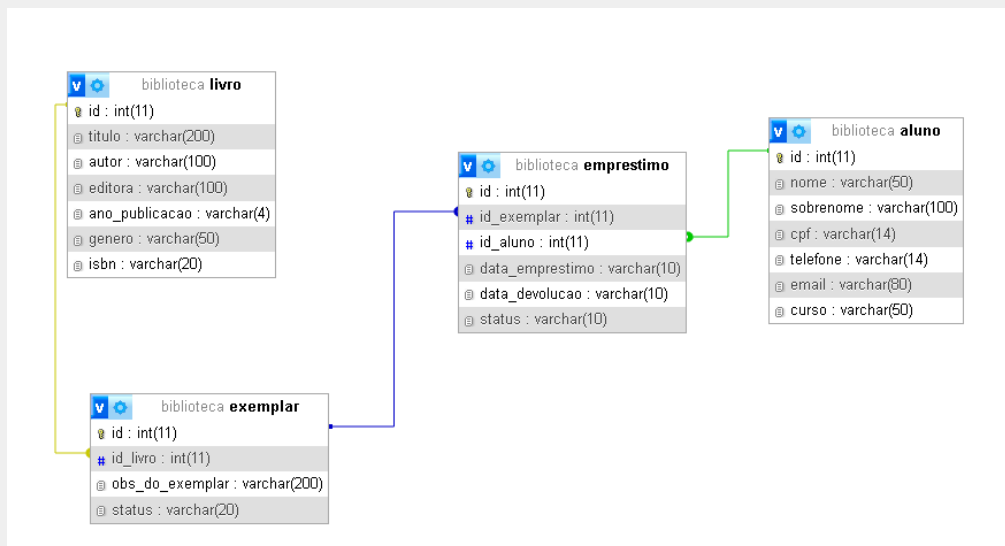
3. Estrutura do Banco de Dados

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.

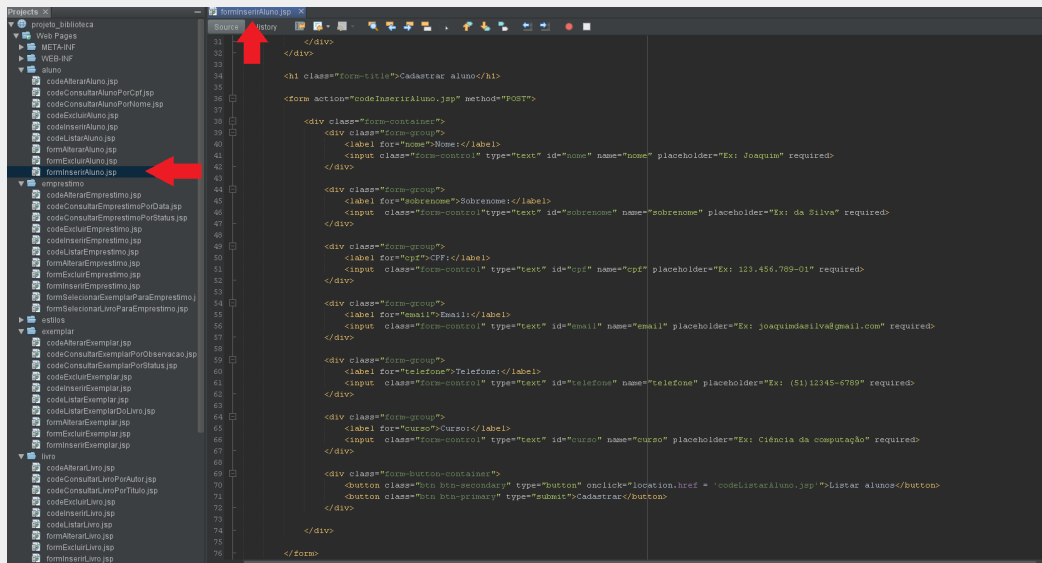
4.1 Páginas criadas

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

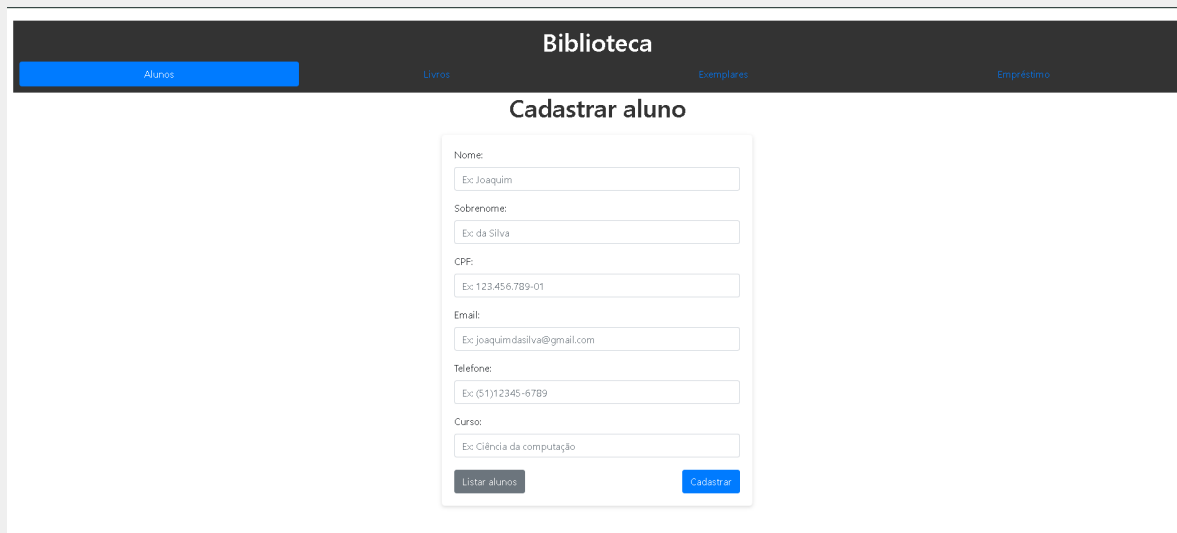
Foram criadas páginas de formulário (onde recebem input do usuário para cadastro, alteração ou exclusão de dados), estas páginas possuem o prefixo “form” no nome.

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

LAYOUT:



Biblioteca

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

Cadastrar aluno

Nome:

Sobrenome:

CPF:

Email:

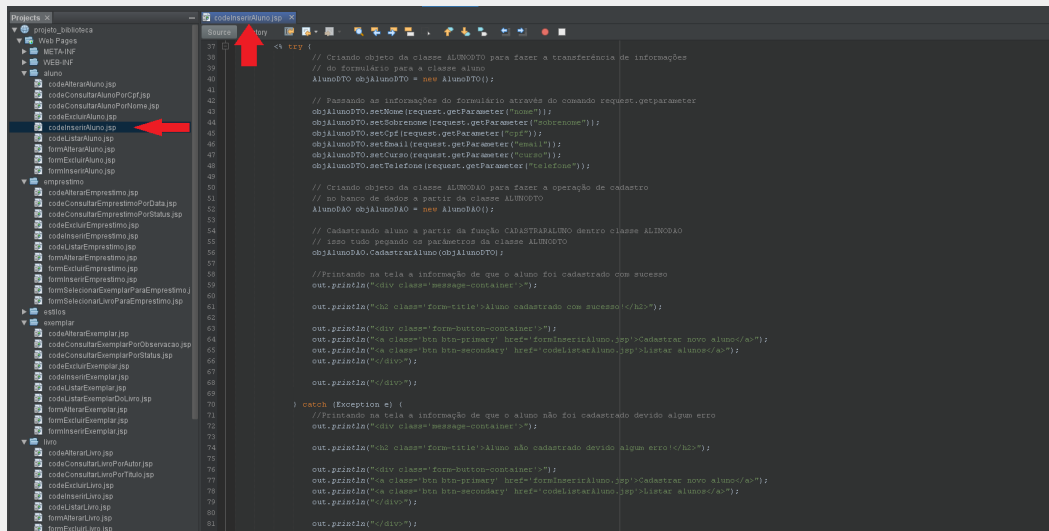
Telefone:

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:

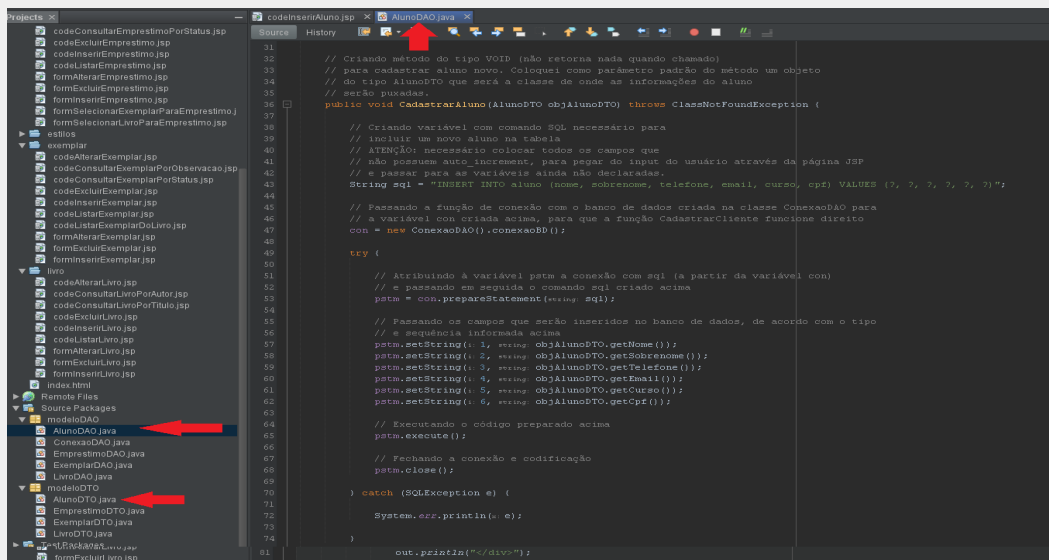


```
1 1 codeEmprestimo.jsp
2
3 // try {
4
5 // Criando objeto da classe ALUNO para fazer a transferência de informações
6 // do formulário para a classe aluno
7 AlunoDTO objAlunoDTO = new AlunoDTO();
8
9 // Passando as informações do formulário através do comando request.getParameter
10 objAlunoDTO.setNome(request.getParameter("nome"));
11 objAlunoDTO.setSobrenome(request.getParameter("sobrenome"));
12 objAlunoDTO.setCpf(request.getParameter("cpf"));
13 objAlunoDTO.setEmail(request.getParameter("email"));
14 objAlunoDTO.setCurso(request.getParameter("curso"));
15 objAlunoDTO.setTelefone(request.getParameter("telefone"));
16
17 // Criando objeto da classe ALUNO para fazer a operação de cadastro
18 // no banco de dados a partir da classe ALUNO
19 AlunoDAO objAlunoDAO = new AlunoDAO();
20
21 // Cadastro de aluno a partir da função CADASTRARALUNO dentro classe ALUNO
22 objAlunoDAO.CadastrarAluno(objAlunoDTO);
23
24 // Printando na tela a informação de que o aluno foi cadastrado com sucesso
25 out.println("div classe= message-container");
26
27 out.println("<div classe= form-title>Aluno cadastrado com sucesso/</div>");
28
29 out.println("<div classe= form-button-container>");
30 out.println("<a classe= btn btn-primary href= formInserirAluno.jsp>Cadastrar novo aluno/a/</a>");
31 out.println("<a classe= btn btn-secondary href= codListarAluno.jsp>Listar alunos/a/</a>");
32 out.println("</div>");
33
34 } catch (Exception e) {
35 // Printando na tela a informação de que o aluno não foi cadastrado devido algum erro
36 out.println("div classe= message-container");
37
38 out.println("<div classe= form-title>Aluno não cadastrado devido algum erro/</div>");
39
40 out.println("<div classe= form-button-container>");
41 out.println("<a classe= btn btn-primary href= formInserirAluno.jsp>Cadastrar novo aluno/a/</a>");
42 out.println("<a classe= btn btn-secondary href= codListarAluno.jsp>Listar alunos/a/</a>");
43 out.println("</div>");
44
45 }
46
47 out.println("</div>");
48 }
```

LAYOUT:



CLASSE JAVA:



```
1 1 AlunoDAO.java
2
3 // try {
4
5 // Criando método do tipo VOID (não retorna nada quando chamado)
6 // para cadastrar aluno novo. Coloquei como parametro padrão do método um objeto
7 // do tipo AlunoDTO que será a classe de onde as informações do aluno
8 // serão puxadas.
9 public void CadastrarAluno(AlunoDTO objAlunoDTO) throws ClassNotFoundException {
10
11 // Criando variável com comando SQL necessário para
12 // incluir um novo aluno na tabela
13 // ATENÇÃO: necessário colocar todos os campos que
14 // não possuem auto_increment, para pegar do input do usuário através da página JSP
15 // e passar para as variáveis ainda não declaradas.
16 String sql = "INSERT INTO aluno (nome, sobrenome, telefone, email, curso, cpf) VALUES (?, ?, ?, ?, ?, ?)";
17
18 // Passando a função de conexão com o banco de dados criada na classe ConexaoDAO para
19 // a variável con criada acima, para que a função CadastrarCliente funcione direito
20 con = new ConexaoDAO().conexaoBD();
21
22 try {
23 // Atribuindo à variável pstmt a conexão com sql (a partir da variável con)
24 // e passando em seguida o comando sql criado acima
25 pstmt = con.prepareStatement(sql);
26
27 // Passando os campos que serão inseridos no banco de dados, de acordo com o tipo
28 // e sequência informada acima
29 pstmt.setString(1, objAlunoDTO.getNome());
30 pstmt.setString(2, objAlunoDTO.getSobrenome());
31 pstmt.setString(3, objAlunoDTO.getTelefone());
32 pstmt.setString(4, objAlunoDTO.getEmail());
33 pstmt.setString(5, objAlunoDTO.getCurso());
34 pstmt.setString(6, objAlunoDTO.getCpf());
35
36 // Executando o código preparado acima
37 pstmt.executeUpdate();
38
39 // Fechando a conexão e codificação
40 pstmt.close();
41
42 } catch (SQLException e) {
43 System.err.println(e);
44 }
45
46 out.println("</div>");
47 }
```

```
Project: codeConsultaAlunoPorNome.jsp
codeConsultaAlunoPorNome.jsp
codeConsultaEmprestimoPorStatus.jsp
codeExcluirEmprestimo.jsp
codeInserirEmprestimo.jsp
codeListarEmprestimo.jsp
formAlterarEmprestimo.jsp
formExcluirEmprestimo.jsp
formInserirEmprestimo.jsp
formSelecionarExemplarParaEmprestimo.jsp
formSelecionarLivroArEmprestimo.jsp
exclui
exemplar
codeAlterarExemplar.jsp
codeConsultaExemplarPorObservacao.jsp
codeConsultaExemplarPorStatus.jsp
codeExcluirExemplar.jsp
codeInserirExemplar.jsp
codeListarExemplar.jsp
codeListarExemplarDoLivro.jsp
formAlterarExemplar.jsp
formExcluirExemplar.jsp
formInserirExemplar.jsp
ex
codeAlterarLivro.jsp
codeConsultaLivroPorTitulo.jsp
codeExcluirLivro.jsp
codeInserirLivro.jsp
codeListarLivro.jsp
formAlterarLivro.jsp
formExcluirLivro.jsp
formInserirLivro.jsp
index.html
Remote Files
Source Packages
modeloDAO
AlunoDAO.java
ConexaoDAO.java
EmprestimoDAO.java
ExemplarDAO.java
LivroDAO.java
modeloDTO
AlunoDTO.java
EmprestimoDTO.java
ExemplarDTO.java
LivroDTO.java
History
// Retornar lista;
}

// Metodo para consultar aluno por nome
public ArrayList<AlunoDTO> consultarAlunoPorNome(String nome) throws ClassNotFoundException {
    String sql = "SELECT * FROM aluno WHERE nome LIKE ?";

    con = new ConexaoDAO().conexaoBD();

    try {
        pstmt = con.prepareStatement(sql);
        pstmt.setString(1, nome + "%");
        rs = pstmt.executeQuery();

        while (rs.next()) {
            AlunoDTO objAlunoDTO = new AlunoDTO();
            objAlunoDTO.setId(rs.getInt("id"));
            objAlunoDTO.setNome(rs.getString("nome"));
            objAlunoDTO.setSobrenome(rs.getString("sobrenome"));
            objAlunoDTO.setTelefone(rs.getString("telefone"));
            objAlunoDTO.setEmail(rs.getString("email"));
            objAlunoDTO.setCidade(rs.getString("cidade"));
            objAlunoDTO.setCpf(rs.getString("cpf"));
            lista.add(objAlunoDTO);
        }

        pstmt.close();
    } catch (SQLException e) {
        System.out.println(e);
    }

    return lista;
}

// Criando metodo para retornar o titulo do livro a partir do ID informado
public String obterNomeDoLivro(int idaluno) throws ClassNotFoundException {
    // Criando variavel para armazenar o titulo do livro
    String nome = "";

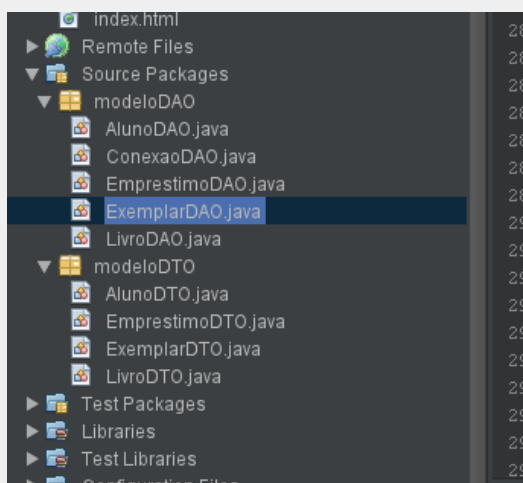
    // Criando comando SQL para extrair o titulo do livro a partir do ID informado
    String sql = "SELECT nome FROM aluno WHERE id = ?";
}
```

4.2 Classes JAVA

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

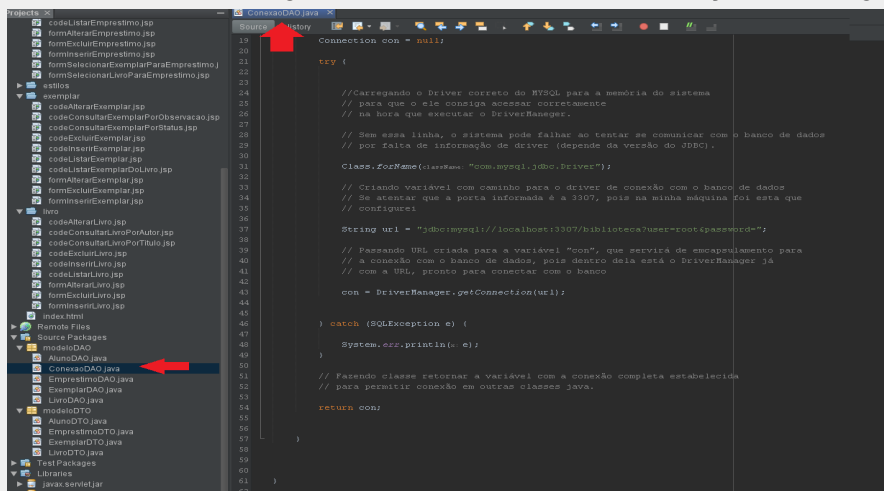
4.2.1 Estrutura de pacotes

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

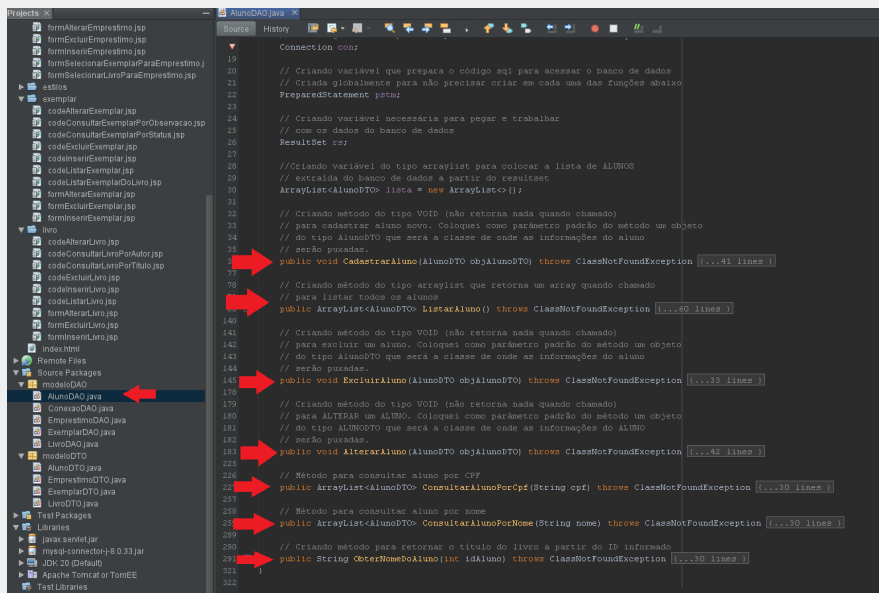
Foi criada uma classe específica para a conexão do banco de dados, com a finalidade de poupar linhas de código e otimizar a leitura e manutenção do código.



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.