Documentação do Projeto: Controle de Orçamentos para uma Empresa

Descrição Geral

Este projeto é um sistema web desenvolvido para gerenciar o cadastro de clientes e os orçamentos relacionados a uma empresa. Ele permite que os usuários executem operações de CRUD (Criar, Ler, Atualizar e Deletar) para gerenciar informações de clientes e orçamentos. A aplicação utiliza o padrão MVC (Model-View-Controller) para separar a lógica de negócios, a camada de apresentação e a camada de controle.

Tecnologias Utilizadas

- **Java** para o backend
- **JSP** com JSTL para a camada de apresentação
- **Servlets** para controle de fluxo
- **MySQL** para o banco de dados
- **Tomcat** como servidor de aplicação
- **JDBC** para acesso ao banco de dados
- **Firebase** para autenticação (caso aplicável)

Estrutura de Diretórios

A estrutura de diretórios é organizada da seguinte maneira:

- **src/controller**: Contém os Servlets para controlar as operações e a lógica de fluxo da aplicação.
- **src/model/dao**: Contém as classes DAO (Data Access Object) para manipulação dos dados no banco de dados.
- **src/model/dto**: Contém as classes DTO (Data Transfer Object), que representam as entidades do sistema.
- **WebContent**: Contém arquivos JSP para a interface do usuário e uma pasta `assets` com arquivos estáticos, como CSS, JavaScript, e imagens.

Modelo de Dados (Banco de Dados MySQL)

Estrutura do Banco de Dados

Banco de Dados: `MeuBanco`

Tabela `Clientes`

Armazena as informações dos clientes.

Coluna	Tipo	Descrição
id	INT	Identificador único (PK)
nome	VARCHAR(100)	Nome do cliente
idade	INT	Idade do cliente
sexo	ENUM	Gênero do cliente
contato	VARCHAR(50)	Informação de contato

Tabela `Orcamento`

Armazena informações sobre os orçamentos.

Coluna	Tipo	Descrição
id	INT	Identificador único (PK)
projeto	VARCHAR(100)	Nome do projeto
valor	DECIMAL(10,2)	Valor do orçamento
cliente_id	INT	ID do cliente relacionado (FK)
status	ENUM	Status do orçamento (Pendente, Aprovado, Rejeitado)

Componentes

1. DTO (Data Transfer Object)

ClienteDTO

Classe `ClienteDTO` representa os dados de um cliente.

java
Copiar código
```
public class ClienteDTO {
    private int id;
    private String nome;
```

```
    private int idade;
    private String sexo;
    private String contato;

    // Métodos getter e setter
}
```

2. DAO (Data Access Object)

ClienteDAO

Classe `ClienteDAO` contém os métodos para realizar operações de CRUD no banco de dados para os dados de `Clientes`.

Principais métodos:

- `inserirCliente(ClienteDTO cliente)`: Insere um novo cliente no banco de dados.
- `listarClientes()`: Retorna uma lista com todos os clientes.
- `buscarClientePorId(int id)`: Retorna as informações de um cliente pelo ID.
- `atualizarCliente(ClienteDTO cliente)`: Atualiza as informações de um cliente existente.
- `excluirCliente(int id)`: Exclui um cliente pelo ID.

---

3. Controller (Servlets)

Para as operações de CRUD, criamos os seguintes Servlets:

CadastrarClienteServlet

- Rota: `/cadastrarCliente`
- Função: Recebe dados do formulário para cadastrar um novo cliente.

ListarClientesServlet

- Rota: `/listarClientes`
- Função: Recupera todos os clientes do banco de dados e envia para a exibição na página JSP.

AtualizarClienteServlet

- Rota: `/atualizarCliente`

- Função: Recebe dados do formulário para atualizar as informações de um cliente existente.

ExcluirClienteServlet

- Rota: `/excluirCliente`
- Função: Exclui um cliente com base no ID passado.

---

4. Páginas JSP

cliente.jsp

Página JSP principal que exibe todas as operações de CRUD para `Clientes`. Utiliza JSTL para controle de fluxo e iteração.

Principais componentes:

- Formulário para cadastro e atualização de clientes.
- Tabela que lista todos os clientes.
- Botões para editar e excluir clientes.

Importações e Tags Importantes

Para usar JSTL, incluímos o seguinte import no início da página JSP:

jsp
Copiar código
```
<%@ taglib uri="http://xmlns.jcp.org/jsp/jstl/core" prefix="c" %>
```

Esse import permite o uso de tags JSTL como `<c:forEach>` e `<c:if>`, tornando o código JSP mais limpo e organizado.

Configurações Adicionais

Configuração do Banco de Dados

No arquivo de configuração do projeto (pode ser um `context.xml` ou no código do `Servlet`), configure a conexão com o banco de dados MySQL:

java
Copiar código

```
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/MeuBanco", "usuario", "senha");
```

Dependências Necessárias

Certifique-se de incluir as bibliotecas necessárias (JSTL, MySQL Connector) no classpath, especialmente em `WEB-INF/lib`, se não estiver usando Maven.

Estruturação do Projeto para Deploy no Tomcat

O projeto deve ser empacotado como um `.war` e colocado na pasta `webapps` do Tomcat para ser executado no servidor.

Funcionamento Geral

1. O usuário acessa a página JSP (`cliente.jsp`) para gerenciar clientes.
2. As ações (cadastrar, atualizar, excluir) disparam requisições aos `Servlets` correspondentes, que manipulam os dados via DAO.
3. Os `Servlets` redirecionam ou despacham para a página JSP com a lista atualizada de clientes.

Conclusão

Este sistema de Controle de Orçamentos permite à empresa gerenciar facilmente os clientes e orçamentos associados. Ele segue o padrão MVC, o que facilita a manutenção e expansão futura. Com as dependências e configurações adequadas, o projeto é fácil de implantar e escalável para novos módulos ou funcionalidades adicionais.