Desenvolvimento de Aplicação Web Java EE para Cadastro de Produtos

Edson Victor Miranda de Oliveira – 2024 0836 7775

Polo Parangaba - Fortaleza/CE

Backend Java Cloud – 2025/3

3.2. Objetivo da Prática

O objetivo desta prática foi desenvolver uma aplicação Web corporativa utilizando a plataforma Java EE, aplicando o padrão de arquitetura em camadas (apresentação, controle, negócio e persistência).

A aplicação implementa o cadastro de produtos, permitindo registrar, listar, atualizar e excluir informações em um banco de dados relacional por meio do uso de JPA e EJB.

Também foram explorados os conceitos de Servlets, JSPs e o padrão Front Controller, além da utilização do Bootstrap para melhorar o design e a responsividade da interface.

Todo o código foi versionado em um repositório no Git, e o link do repositório está incluído neste relatório, conforme orientações da Missão Prática.

(Nesse parágrafo você pode colocar o link do GitHub no final.)

3.3. 1º Procedimento | Camadas de Persistência e Controle

Responda os itens a–e do template. Sugestão de texto:

a. Como é organizado um projeto corporativo no NetBeans?

No NetBeans, um projeto corporativo é organizado em módulos e camadas bem definidas. As classes de entidade e persistência (JPA) ficam geralmente em pacotes específicos dentro do projeto EJB, enquanto as classes de negócio (Session Beans) ficam em outro pacote, separando regras de negócio do acesso a dados. Já o módulo Web concentra Servlets, JSPs e recursos estáticos. Essa separação facilita a manutenção, o reuso e o trabalho em equipe.

b. Qual o papel das tecnologias JPA e EJB na construção de um aplicativo Web Java?

A JPA é responsável pelo mapeamento objeto–relacional, permitindo que as entidades Java sejam persistidas em um banco de dados de forma transparente, por meio de anotações e de uma unidade de persistência.
Os EJBs (Session Beans) encapsulam a lógica de negócio, oferecendo serviços como

transações, segurança, pooling e injeção de dependências. Na prática, o Servlet chama os Session Beans, que por sua vez utilizam a JPA para acessar e manipular os dados.

c. Como o NetBeans viabiliza a melhoria de produtividade ao lidar com JPA e EJB?

O NetBeans oferece wizards para criação de entidades JPA a partir do banco de dados, geração automática de Session Beans e até de facades para CRUD. Além disso, o IDE auxilia na configuração da persistence.xml, no deploy do projeto no servidor de aplicação e na criação de Servlets, reduzindo o trabalho manual e o risco de erros.

d. O que são Servlets e como o NetBeans oferece suporte?

Servlets são componentes Java que processam requisições HTTP no lado do servidor, funcionando como controladores da aplicação Web. O NetBeans facilita sua criação por meio de assistentes que geram a classe básica, mapeamento em web.xml (ou por anotação) e integração com o servidor. Também permite depuração passo a passo e visualização das requisições.

e. Como é feita a comunicação entre Servlets e Session Beans?

A comunicação é feita por injeção de dependência ou lookup. O Servlet recebe a requisição do usuário, obtém a referência para o Session Bean (por exemplo, via @EJB) e chama seus métodos de negócio. O Session Bean, por sua vez, utiliza a JPA para acessar o banco, retornando ao Servlet o resultado para ser encaminhado à camada de visão (JSP).

3.4. 2º Procedimento | Interface Cadastral com Servlet e JSPs

a. Como funciona o padrão Front Controller na arquitetura MVC?

No padrão Front Controller, todas as requisições passam primeiro por um único Servlet controlador, que centraliza o fluxo da aplicação. Esse Servlet analisa os parâmetros da requisição, decide qual ação executar (cadastrar, listar, editar, etc.) e depois encaminha a resposta para a JSP apropriada.
Na arquitetura MVC, o Front Controller representa o Controller, os EJBs e entidades representam o Model, e as JSPs representam a View.

b. Diferenças e semelhanças entre Servlets e JSPs

Ambos fazem parte da camada de apresentação e trabalham com requisições HTTP.

- **Servlets são classes Java, mais adequadas para controle de fluxo e regras de navegação.**
- **JSPs são páginas voltadas para apresentação, permitindo misturar HTML com tags JSP/EL, o que facilita a construção da interface.**
  A combinação comum é: Servlet processa a lógica, define atributos na requisição e encaminha para uma JSP renderizar a resposta.

c. Diferença entre redirecionamento simples e forward do RequestDispatcher

- **O redirecionamento simples (response.sendRedirect) envia uma nova resposta ao navegador, que faz outra requisição para a URL indicada. O endereço muda na barra e envolve duas requisições HTTP.**
- **O forward do RequestDispatcher ocorre no lado do servidor: o controle é repassado internamente para outro recurso (JSP ou Servlet) sem que o navegador saiba, mantendo a mesma requisição.**
  Parâmetros são os valores enviados pelo cliente (query string, formulários) e atributos são objetos adicionados no HttpServletRequest pelo servidor para serem compartilhados entre Servlets e JSPs.

3.5. 3º Procedimento | Melhorando o Design da Interface

a. Como o Bootstrap é utilizado?

O Bootstrap é incorporado ao projeto por meio de links CSS e JavaScript nas páginas JSP. A partir daí, são usados componentes prontos (botões, formulários, tabelas, cards) e classes utilitárias para organizar o layout da tela de cadastro e listagem de produtos, deixando o visual mais moderno e padronizado.

b. Por que o Bootstrap garante independência estrutural do HTML?

Porque ele trabalha com classes CSS aplicadas a elementos HTML simples (div, table, button, etc.). A estrutura da página permanece baseada em HTML padrão, enquanto o comportamento visual e o layout são definidos pelas classes do Bootstrap. Assim, o desenvolvedor altera o visual apenas trocando ou combinando classes, sem precisar reescrever toda a estrutura.

c. Relação entre Bootstrap e responsividade da página

O Bootstrap utiliza um sistema de grid responsivo e media queries que ajustam o layout automaticamente para diferentes tamanhos de tela (desktop, tablet, celular). Ao definir colunas e containers com as classes adequadas, a mesma página se adapta a vários dispositivos, melhorando a experiência do usuário sem necessidade de várias versões do site.

3.6. Conclusão

A Missão Prática permitiu integrar diversos conceitos importantes da plataforma Java EE, desde a organização de um projeto corporativo até o consumo de dados via JPA e EJB, passando pela camada Web com Servlets, JSPs e Bootstrap.

Como resultado, foi possível construir uma aplicação funcional de cadastro de produtos, com interface amigável e código versionado em Git.

Entre as principais dificuldades estiveram a configuração inicial do servidor de aplicação e da unidade de persistência, além da compreensão da comunicação entre Servlets e Session Beans. Por outro lado, o uso do NetBeans, dos wizards de JPA/EJB e do Bootstrap ajudou a aumentar a produtividade.

De forma geral, a prática consolidou o entendimento da arquitetura em camadas e do uso de ferramentas corporativas para desenvolvimento Web em Java, preparando o aluno para projetos mais complexos na área de backend Java em ambiente cloud.