



**Estácio**

# Universidade Estácio de Sá

3º período em Desenvolvimento Full stack

Matrícula: 202302891292

Aluno: Luiz Fabrício Mello Ferreira

## Vamos Integrar Sistemas

## 1º Procedimento | Camadas de Persistência e Controle

### Objetivo da Prática

- Implementar persistência com base em JPA.
- Implementar regras de negócio na plataforma JEE, através de EJBs.
- Implementar sistema cadastral Web com base em Servlets e JSPs.
- Utilizar a biblioteca Bootstrap para melhoria do design.
- No final do exercício, o aluno terá criado todos os elementos necessários para
- exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para
- lidar com contextos reais de aplicação.

**Códigos solicitados estão no repositório:**

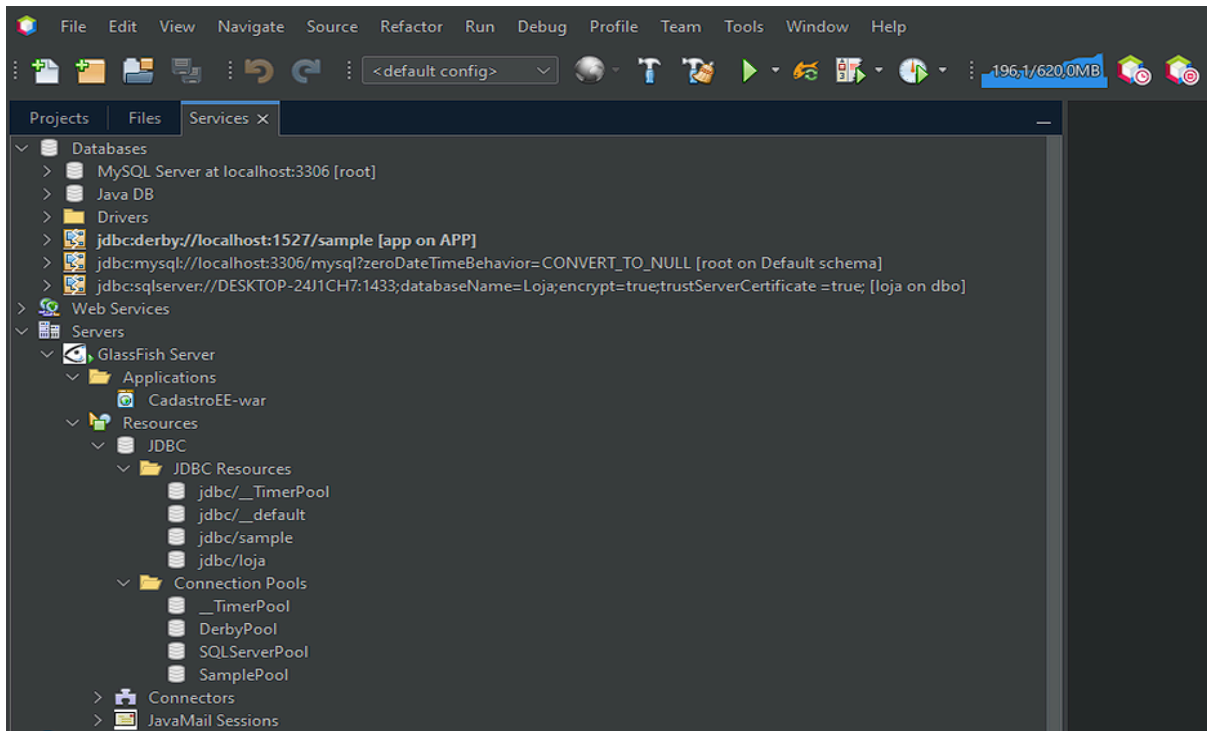
<https://github.com/luizfabriorex/UNESA/tree/main/terceiroPeriodo/pratica-4>

### Resultado da execução dos códigos:

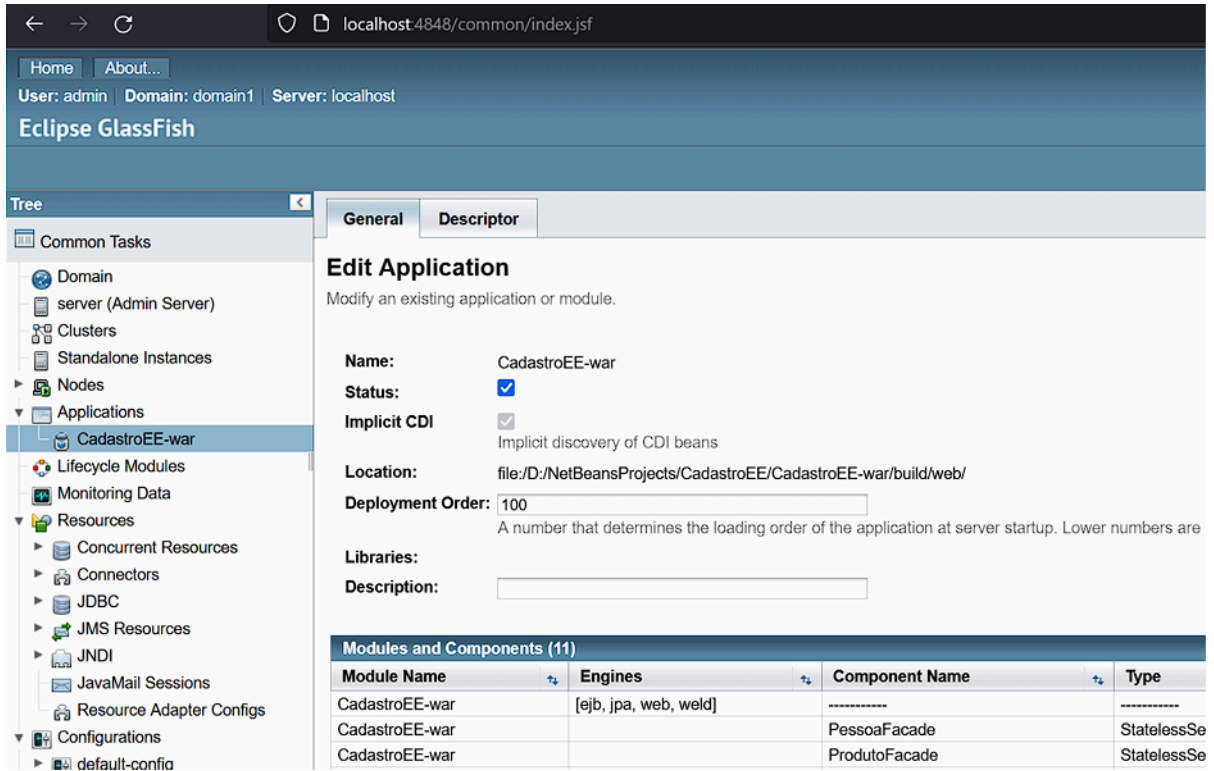
A configuração do ambiente e de todo o procedimento foi bem sucedidos, mas honestamente não é trivial, com alguns detalhes que requerem bastante atenção. Por exemplo, as strings de parâmetros de GlassFish contidas no enunciado da missão não estão totalmente corretas; foi necessário que o tutor da disciplina corrigisse no fórum.

Mesmo com o passo-a-passo detalhado, iniciantes que nunca entraram em contato com essas tecnologias (NetBeans, GlassFish, MS SQL Server Studio, JDBC) certamente terão dificuldades.

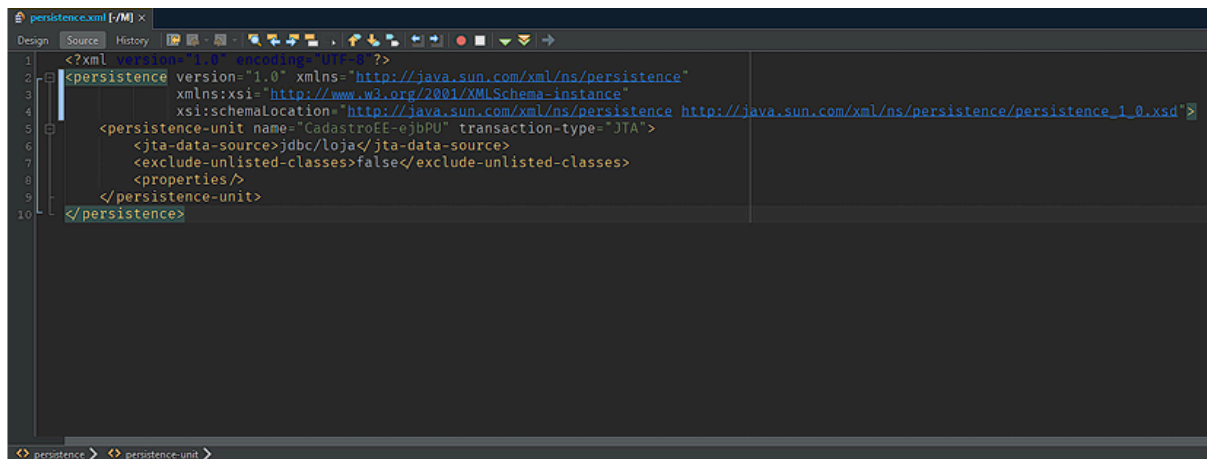
Contudo, a seguir seguem as figuras para fins de comprovação do sucesso da missão. Em especial, da correta configuração do GlassFish (fig. 1), da aplicação configurada via NetBeans e exibida na interface web do GlassFish (fig. 2), a configuração do arquivo persistence.xml (fig. 3) e a exibição da execução do ServletProduto (fig. 4), conforme solicitado.



**Figura 1.** Serviços configurados no NetBeans: JDBC, GlassFish, jdbc/Loja, SQLServerPool



**Figura 2:** aplicação CadastroEE-war em execução no GlassFish



**Figura 3:** Arquivo de configuração persistence.xml



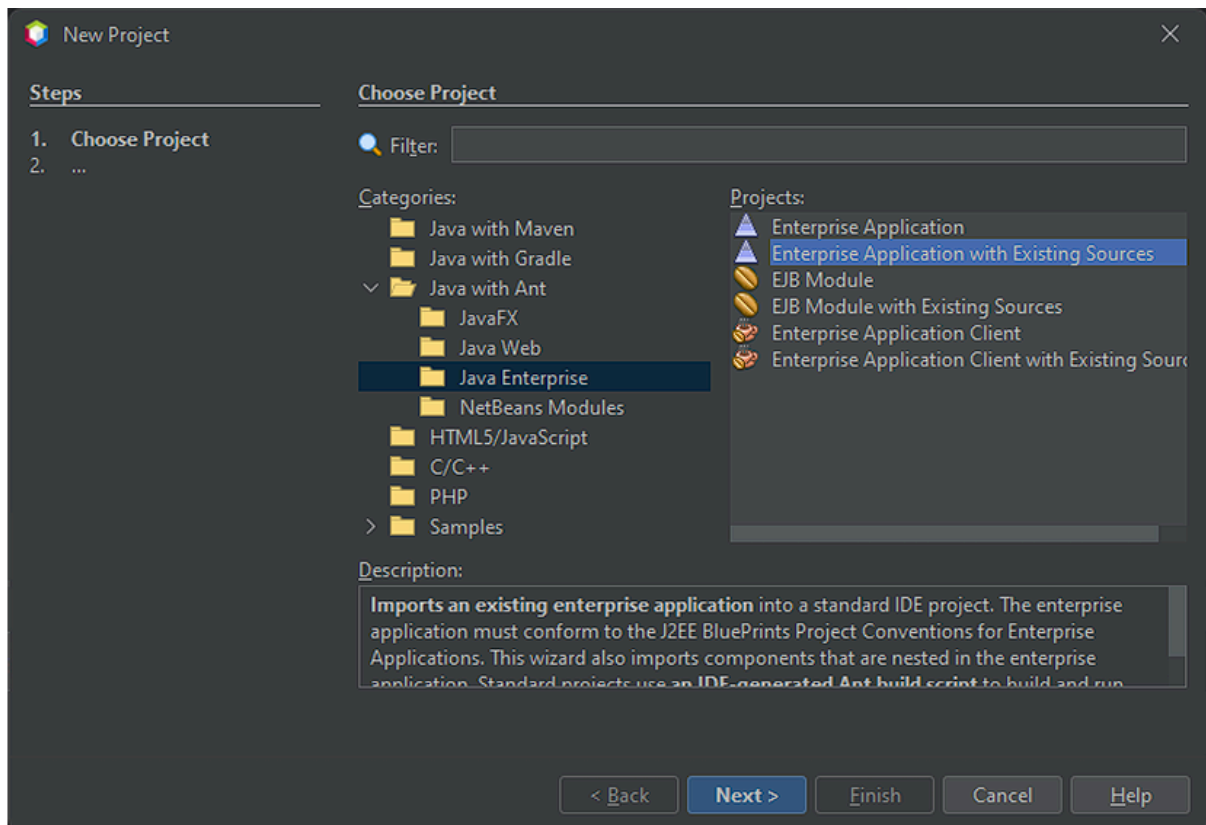
**Figura 4.** ServletProduto em execução.

**Figura 4.** ServletProduto em execução.

## 5. Análise e Conclusão

### A) Como é organizado um projeto corporativo no NetBeans?

Um projeto corporativo (ou Enterprise) no NetBeans segue uma estrutura modular hierárquica, claramente visível ao se criar um novo projeto: escolher uma categoria de gerenciador de pacotes, dentre os mais utilizados na atualidade: Maven, Gradle, Ant (fig. 5); em seguida, escolher o tipo de projeto corporativo mais adequado, conforme mostra a figura abaixo:



**Figura 5.** Criação de um novo projeto corporativo no NetBeans.

Projetos corporativos web no NetBeans, mesmo que sejam de diferentes categorias e tipos, seguem um padrão comum de pastas: código-fonte (source), bibliotecas ou dependências (libraries), configurações (configuration ou WEB-INF), páginas web ou servlets (web pages, war).

Nesta Missão Prática, foi escolhida a categoria “Java with Ant”, seguida da sub-categoria “Java Enterprise”, com tipo de projeto “Enterprise Application with Existing Sources”, o qual é gerado a partir de um banco de dados existente, como resultado: além da pasta principal Enterprise Application, são criados 2 módulos com diferentes funções: um módulo EJB (Cadastro-ejb) e um módulo de aplicação web (Cadastro-war), gerenciados separadamente e integrados para apresentar a total funcionalidade do sistema. O NetBeans facilita a organização desses módulos, com o uso de ferramentas e assistentes de configuração, construção e implantação de maneira eficiente.

B) Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?

As tecnologias JPA (Jakarta Persistence API [1], anteriormente Java Persistence API) e EJB (Enterprise JavaBeans) desempenham papéis importantes na construção de aplicativos para a plataforma web no ambiente Java.

O JPA é uma API (Application Program Interface) de persistência de dados padrão do Java, o qual permite aos desenvolvedores mapear objetos Java para tabelas em um banco de dados relacional. Com JPA, é possível escrever consultas em linguagem Java, as quais são traduzidas para consultas SQL pelo provedor de persistência. Isso simplifica o processo de interação com o banco de dados e torna o código mais portátil entre diferentes provedores de banco de dados. O JPA é frequentemente usado para lidar com operações de banco de dados em aplicativos web Java.

O EJB é um componente de servidor do Java EE (Enterprise Edition) que simplifica o desenvolvimento de aplicativos corporativos. O EJB oferece uma maneira de encapsular a lógica de negócios em componentes reutilizáveis que podem ser implantados em um servidor de aplicativos Java EE, tais como o Eclipse GlassFish ou o Apache Tomcat.

Portanto, em uma aplicação web Java, pode-se utilizar o JPA para mapear e persistir objetos Java em um banco de dados relacional, e o EJB para implementar a lógica de negócios do aplicativo, tais como autenticação de usuário, processamento de requisições e assim por diante. Na prática, essas tecnologias combinadas podem ajudar a criar aplicativos web robustos e escaláveis no ambiente Java.

Um exemplo de uso: há diferentes tipos de EJBs, como EJBs de sessão, utilizados para implementar a lógica de negócios da aplicação, e EJBs de entidade, utilizados para representar entidades persistentes, geralmente mapeadas com JPA.

C) Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?

NetBeans possui **suporte integrado para JPA e EJB**, como a inclusão de assistentes para criar entidades JPA a partir de tabelas de banco de dados existentes, a geração automática de código para EJBs, a integração com os

servidores de aplicativos Java EE para implantar e testar aplicativos que utilizam essas tecnologias.

NetBeans também possui **ferramentas de mapeamento de entidades JPA**, as quais permitem criar e editar facilmente os mapeamentos entre classes Java e as tabelas de banco de dados com uso do JPA. Isso simplifica o processo de definição de como objetos Java são persistidos no banco de dados, o que resulta em economia de tempo e redução de erros.

NetBeans conta com **geração automática de código para entidades JPA, sessões EJB** e outros componentes, com base em modelos predefinidos e nas configurações do projeto. Isso acelera o desenvolvimento, a fim de que desenvolvedores se concentrem mais na lógica de negócios, ao invés de concentrar esforços na escrita de código repetitivo de infraestrutura.

NetBeans é bastante hábil ao realizar **integração com ferramentas de bancos de dados**, as quais permitem aos desenvolvedores visualizar, modificar esquemas de banco de dados, executar consultas SQL e realizar outras tarefas relacionadas ao banco de dados diretamente na IDE. Isso simplifica o processo de desenvolvimento de aplicativos que utilizam JPA para acesso ao banco de dados.

Finalmente, o NetBeans possui **ferramentas avançadas de debug e testes**, essenciais para desenvolvedores identificarem e corrigirem problemas de desempenho em aplicativos que utilizam as tecnologias JPA e EJB. Isso pode ser especialmente útil ao otimizar consultas de banco de dados e tornar mais eficiente o uso de recursos do servidor.

D) O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?

Servlets são componentes em forma de classes Java, usadas para estender a capacidade de servidores web hospedar aplicativos web dinâmicos; fornecem uma maneira de responder a solicitações HTTP, dinamicamente, com base nos dados enviados pelo cliente, seja através de formulários (form) web ou parâmetros em URL. Servlets fazem parte da especificação Java EE e são comumente usados para lidar com a lógica de controle em aplicativos web Java.

NetBeans oferece um suporte robusto para a construção e o desenvolvimento de servlets em projetos web, como assistentes e modelos, os quais permitem montar automaticamente o esqueleto do servlet com os métodos `doGet()` e `doPost()`, ou então manualmente criar um servlet a partir da herança da classe `javax.servlet.http.HttpServlet`.

NetBeans também gerencia automaticamente o arquivo de configurações `web.xml`, usado para mapear servlets para URLs específicas e configurar outras propriedades do servlet.

NetBeans oferece suporte completo para debug de servlets, o qual permite adicionar breakpoints no código, executar a aplicação em modo de depuração para identificar e corrigir problemas facilmente.

NetBeans integra-se facilmente com servidores de aplicações Java EE, tais como Apache Tomcat e Eclipse GlassFish, o que permite implantar e testar servlets a partir da IDE.

NetBeans também fornece assistência ao desenvolvimento com sugestões de código, realce de sintaxe, refatoração de código e outras ferramentas que ajudam a aumentar a produtividade.

E) Como é feita a comunicação entre os Servlets e os Session Beans do pool de EJBs?

A comunicação entre Servlets e Session Beans em um pool de EJBs é realizada através de um mecanismo chamado **JNDI (Java Naming and Directory Interface)**. Basicamente, os Servlets utilizam JNDI para localizar e referenciar Session Beans disponíveis no pool; após a localização, os Servlets podem invocar métodos nos Session Beans como se realizassem chamadas a métodos em objetos locais. Essa abordagem permite que Servlets, que geralmente gerenciam solicitações de usuários via HTTP, interajam com lógicas de negócios complexas encapsuladas nos Session Beans, assim promovem uma eficiente separação entre camada de apresentação e lógica de negócios.



## 2º Procedimento | Interface Cadastral com Servlet e JSP

### Objetivo da Prática

- Implementar persistência com base em JPA.
- Implementar regras de negócio na plataforma JEE, através de EJBs.
- Implementar sistema cadastral Web com base em Servlets e JSPs.
- Utilizar a biblioteca Bootstrap para melhoria do design.
- No final do exercício, o aluno terá criado todos os elementos necessários para exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para lidar com contextos reais de aplicação.

### Códigos solicitados estão no repositório:

<https://github.com/luizfabriciorex/UNESA/tree/main/terceiroPeriodo/pratica-4>

### Resultado da execução dos códigos:

O 2º procedimento é continuação do 1º procedimento, com a diferença que é criado um novo servlet denominado ServletProdutoFC (fig. 6), que utiliza o padrão FC (Front Controller), com a capacidade de exibir a lista de produtos, cadastrar, alterar e excluir produtos, armazenados em banco de dados, através de parâmetros de “ação” na URL (“incluir”, “alterar”, “excluir”).

As interfaces gráficas JSP (Java Server Page) (fig. 7), conforme solicitado neste 2º procedimento, utilizam basicamente HTML, sem uso de bibliotecas CSS, como mostrado na fig. 8. Essas interfaces JSP são templates escritos em HTML salvos em arquivos de extensão .jsp, com uso adequado e posicionado de tags <% %>, onde os dados da aplicação são corretamente inseridos e exibidos.

```

16 import jakarta.servlet.http.HttpServletRequest;
17 import jakarta.servlet.http.HttpServletResponse;
18
19 @WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})
20 public class ServletProdutoFC extends HttpServlet {
21
22     @EJB
23     private ProdutoFacadeLocal facade;
24
25     @Override
26     protected void doGet(HttpServletRequest request, HttpServletResponse response)
27         throws ServletException, IOException {
28         String acao = request.getParameter(string:"acao");
29         String destino = handleGetAction(acao, request);
30         dispatchRequest(request, response, destino);
31     }
32
33     @Override
34     protected void doPost(HttpServletRequest request, HttpServletResponse response)
35         throws ServletException, IOException {
36         String acao = request.getParameter(string:"acao");
37         acao = acao == null || acao.isEmpty() ? " " : acao;
38         String destino = handlePostAction(acao, request);
39         dispatchRequest(request, response, destino);
40     }
41
42     private String handleGetAction(String acao, HttpServletRequest request) {
43         switch (acao) {
44             case "formIncluir":
45                 return "ProdutoDados.jsp";
46             case "excluir":
47                 return handleExcluir(request);
48             case "formAlterar":
49                 return handleFormAlterar(request);
50             default:
51                 return handleListarProdutos(request);
52         }
53     }
54
55     private String handlePostAction(String acao, HttpServletRequest request) {
56         switch (acao) {
57             case "incluir":
58                 return handleIncluir(request);
59         }
60     }
61 }

```

Figura 6. Trecho do código do ServletProdutoFC.

```

64 <table class="table table-striped table-bordered table-responsive">
65 <thead>
66 <tr class="table-dark">
67 <th>ID</th>
68 <th>Produto</th>
69 <th>Quantidade</th>
70 <th>Preço</th>
71 <th>Ações</th>
72 </tr>
73 </thead>
74
75 <tbody>
76 <!-- Loop for products -->
77
78 <!-- Empty state message -->
79
80 </tbody>
81 </table>
82
83 <div class="text-end mb-3">
84 <a class="btn btn-primary" href="ServletProdutoFC?acao=formIncluir">Cadastrar Produto</a>
85 </div>
86
87 </div>
88
89 </div>
90
91 </div>
92
93 </div>
94
95 </div>
96
97 </div>
98
99 </div>
100
101 </div>
102
103 </div>
104
105 </div>
106
107 </div>
108
109 </div>
110
111 </div>
112
113 </div>
114
115 </div>
116
117 </div>
118
119 </div>
120
121 </div>
122
123 </div>
124
125 </div>
126
127 </div>
128
129 </div>
130
131 </div>
132
133 </div>
134
135 </div>
136
137 </div>
138
139 </div>
140
141 </div>
142
143 </div>
144
145 </div>
146
147 </div>
148
149 </div>
150
151 </div>
152
153 </div>
154
155 </div>
156
157 </div>
158
159 </div>
160
161 </div>
162
163 </div>
164
165 </div>
166
167 </div>
168
169 </div>
170
171 </div>
172
173 </div>
174
175 </div>
176
177 </div>
178
179 </div>
180
181 </div>
182
183 </div>
184
185 </div>
186
187 </div>
188
189 </div>
190
191 </div>
192
193 </div>
194
195 </div>
196
197 </div>
198
199 </div>
200
201 </div>
202
203 </div>
204
205 </div>
206
207 </div>
208
209 </div>
210
211 </div>
212
213 </div>
214
215 </div>
216
217 </div>
218
219 </div>
220
221 </div>
222
223 </div>
224
225 </div>
226
227 </div>
228
229 </div>
230
231 </div>
232
233 </div>
234
235 </div>
236
237 </div>
238
239 </div>
240
241 </div>
242
243 </div>
244
245 </div>
246
247 </div>
248
249 </div>
250
251 </div>
252
253 </div>
254
255 </div>
256
257 </div>
258
259 </div>
260
261 </div>
262
263 </div>
264
265 </div>
266
267 </div>
268
269 </div>
270
271 </div>
272
273 </div>
274
275 </div>
276
277 </div>
278
279 </div>
280
281 </div>
282
283 </div>
284
285 </div>
286
287 </div>
288
289 </div>
290
291 </div>
292
293 </div>
294
295 </div>
296
297 </div>
298
299 </div>
300
301 </div>
302
303 </div>
304
305 </div>
306
307 </div>
308
309 </div>
310
311 </div>
312
313 </div>
314
315 </div>
316
317 </div>
318
319 </div>
320
321 </div>
322
323 </div>
324
325 </div>
326
327 </div>
328
329 </div>
330
331 </div>
332
333 </div>
334
335 </div>
336
337 </div>
338
339 </div>
340
341 </div>
342
343 </div>
344
345 </div>
346
347 </div>
348
349 </div>
350
351 </div>
352
353 </div>
354
355 </div>
356
357 </div>
358
359 </div>
360
361 </div>
362
363 </div>
364
365 </div>
366
367 </div>
368
369 </div>
370
371 </div>
372
373 </div>
374
375 </div>
376
377 </div>
378
379 </div>
380
381 </div>
382
383 </div>
384
385 </div>
386
387 </div>
388
389 </div>
390
391 </div>
392
393 </div>
394
395 </div>
396
397 </div>
398
399 </div>
400
401 </div>
402
403 </div>
404
405 </div>
406
407 </div>
408
409 </div>
410
411 </div>
412
413 </div>
414
415 </div>
416
417 </div>
418
419 </div>
420
421 </div>
422
423 </div>
424
425 </div>
426
427 </div>
428
429 </div>
430
431 </div>
432
433 </div>
434
435 </div>
436
437 </div>
438
439 </div>
440
441 </div>
442
443 </div>
444
445 </div>
446
447 </div>
448
449 </div>
450
451 </div>
452
453 </div>
454
455 </div>
456
457 </div>
458
459 </div>
460
461 </div>
462
463 </div>
464
465 </div>
466
467 </div>
468
469 </div>
470
471 </div>
472
473 </div>
474
475 </div>
476
477 </div>
478
479 </div>
480
481 </div>
482
483 </div>
484
485 </div>
486
487 </div>
488
489 </div>
490
491 </div>
492
493 </div>
494
495 </div>
496
497 </div>
498
499 </div>
500
501 </div>
502
503 </div>
504
505 </div>
506
507 </div>
508
509 </div>
510
511 </div>
512
513 </div>
514
515 </div>
516
517 </div>
518
519 </div>
520
521 </div>
522
523 </div>
524
525 </div>
526
527 </div>
528
529 </div>
530
531 </div>
532
533 </div>
534
535 </div>
536
537 </div>
538
539 </div>
540
541 </div>
542
543 </div>
544
545 </div>
546
547 </div>
548
549 </div>
550
551 </div>
552
553 </div>
554
555 </div>
556
557 </div>
558
559 </div>
560
561 </div>
562
563 </div>
564
565 </div>
566
567 </div>
568
569 </div>
570
571 </div>
572
573 </div>
574
575 </div>
576
577 </div>
578
579 </div>
580
581 </div>
582
583 </div>
584
585 </div>
586
587 </div>
588
589 </div>
590
591 </div>
592
593 </div>
594
595 </div>
596
597 </div>
598
599 </div>
600
601 </div>
602
603 </div>
604
605 </div>
606
607 </div>
608
609 </div>
610
611 </div>
612
613 </div>
614
615 </div>
616
617 </div>
618
619 </div>
620
621 </div>
622
623 </div>
624
625 </div>
626
627 </div>
628
629 </div>
630
631 </div>
632
633 </div>
634
635 </div>
636
637 </div>
638
639 </div>
640
641 </div>
642
643 </div>
644
645 </div>
646
647 </div>
648
649 </div>
650
651 </div>
652
653 </div>
654
655 </div>
656
657 </div>
658
659 </div>
660
661 </div>
662
663 </div>
664
665 </div>
666
667 </div>
668
669 </div>
670
671 </div>
672
673 </div>
674
675 </div>
676
677 </div>
678
679 </div>
680
681 </div>
682
683 </div>
684
685 </div>
686
687 </div>
688
689 </div>
690
691 </div>
692
693 </div>
694
695 </div>
696
697 </div>
698
699 </div>
700
701 </div>
702
703 </div>
704
705 </div>
706
707 </div>
708
709 </div>
710
711 </div>
712
713 </div>
714
715 </div>
716
717 </div>
718
719 </div>
720
721 </div>
722
723 </div>
724
725 </div>
726
727 </div>
728
729 </div>
730
731 </div>
732
733 </div>
734
735 </div>
736
737 </div>
738
739 </div>
740
741 </div>
742
743 </div>
744
745 </div>
746
747 </div>
748
749 </div>
750
751 </div>
752
753 </div>
754
755 </div>
756
757 </div>
758
759 </div>
760
761 </div>
762
763 </div>
764
765 </div>
766
767 </div>
768
769 </div>
770
771 </div>
772
773 </div>
774
775 </div>
776
777 </div>
778
779 </div>
780
781 </div>
782
783 </div>
784
785 </div>
786
787 </div>
788
789 </div>
790
791 </div>
792
793 </div>
794
795 </div>
796
797 </div>
798
799 </div>
800
801 </div>
802
803 </div>
804
805 </div>
806
807 </div>
808
809 </div>
810
811 </div>
812
813 </div>
814
815 </div>
816
817 </div>
818
819 </div>
820
821 </div>
822
823 </div>
824
825 </div>
826
827 </div>
828
829 </div>
830
831 </div>
832
833 </div>
834
835 </div>
836
837 </div>
838
839 </div>
840
841 </div>
842
843 </div>
844
845 </div>
846
847 </div>
848
849 </div>
850
851 </div>
852
853 </div>
854
855 </div>
856
857 </div>
858
859 </div>
860
861 </div>
862
863 </div>
864
865 </div>
866
867 </div>
868
869 </div>
870
871 </div>
872
873 </div>
874
875 </div>
876
877 </div>
878
879 </div>
880
881 </div>
882
883 </div>
884
885 </div>
886
887 </div>
888
889 </div>
890
891 </div>
892
893 </div>
894
895 </div>
896
897 </div>
898
899 </div>
900
901 </div>
902
903 </div>
904
905 </div>
906
907 </div>
908
909 </div>
910
911 </div>
912
913 </div>
914
915 </div>
916
917 </div>
918
919 </div>
920
921 </div>
922
923 </div>
924
925 </div>
926
927 </div>
928
929 </div>
930
931 </div>
932
933 </div>
934
935 </div>
936
937 </div>
938
939 </div>
940
941 </div>
942
943 </div>
944
945 </div>
946
947 </div>
948
949 </div>
950
951 </div>
952
953 </div>
954
955 </div>
956
957 </div>
958
959 </div>
960
961 </div>
962
963 </div>
964
965 </div>
966
967 </div>
968
969 </div>
970
971 </div>
972
973 </div>
974
975 </div>
976
977 </div>
978
979 </div>
980
981 </div>
982
983 </div>
984
985 </div>
986
987 </div>
988
989 </div>
990
991 </div>
992
993 </div>
994
995 </div>
996
997 </div>
998
999 </div>

```

Figura 7. Trecho do código JSP responsável pela exibição da lista de produtos.



**Figura 8.** ServletProdutoFC em execução, sem uso de bibliotecas CSS, apenas CSS básico

## 5. Análise e Conclusão

A) Como funciona o padrão Front Controller, e como ele é implementado em um aplicativo Web Java, na arquitetura MVC?

O padrão Front Controller (FC) em aplicações web Java, especialmente na arquitetura MVC (Model-View-Controller), funciona como um controlador centralizado que gerencia todas as solicitações do cliente.

Ao invés de utilizar vários controladores para diferentes tipos de solicitações, o FC atua como um ponto único de entrada, que interpreta as solicitações, delega as tarefas apropriadas para modelos específicos (model); em seguida, seleciona a visualização correta (view) para responder ao cliente.

É comum esse fluxo de interações ser implementado através de um servlet, o qual intercepta todas as solicitações, realiza o processamento necessário ou lógica de negócios e, finalmente, encaminha a resposta para a página JSP ou outra tecnologia de visualização de renderização de páginas. Este padrão auxilia na

manutenção e gerenciamento centralizado das solicitações, de modo a promover a reutilização de código em uma estrutura mais organizada.

B) Quais as diferenças e semelhanças entre Servlets e JSPs?

Servlets e JSP são tecnologias usadas para criar aplicações web em Java, mas possuem abordagens bem distintas.

Servlets são classes Java que permitem gerar código HTML a partir de instruções e comandos em Java; assim, os servlets são mais adequados para a lógica de negócios e processamento de dados.

Por outro lado, JSP são páginas HTML com capacidade de incorporar código Java; logo são mais adequados para a apresentação da interface com o usuário. Ambos, servlets e JSP, são executados no servidor de aplicações (GlassFish, Tomcat), com a possibilidade de interagir com bancos de dados, assim como outras tecnologias Java Enterprise.

Em resumo, enquanto servlets são mais voltados para controle, JSPs são mais adequados para a visualização e, frequentemente, ambos são usados em conjunto para separar a lógica de negócios da interface de usuário.

C) Qual a diferença entre um redirecionamento simples e o uso do método forward, a partir do RequestDispatcher?

Para que servem parâmetros e atributos nos objetos HttpRequest? O redirecionamento simples utiliza o método `response.sendRedirect(url)`, enquanto que o RequestDispatcher utiliza o método `forward`. A principal diferença entre eles está no tratamento da requisição. O redirecionamento simples envia uma resposta ao navegador, para indicar que este deve fazer uma nova requisição para outra URL. Já o `forward` encaminha a requisição atual para outro recurso no servidor sem informar ao cliente, e mantém a URL original.

Os parâmetros e atributos em objetos HttpRequest são fundamentais para passar informações entre cliente e servidor, ou entre diferentes partes do servidor. Parâmetros são tipicamente usados para enviar dados de formulários ou de solicitações de URL, enquanto atributos são usados para manter dados durante a

vida útil de uma requisição ou sessão, de modo a permitir a comunicação entre diferentes componentes do servidor.

### 3º Procedimento | Melhorando o Design da Interface

#### Objetivo da Prática

- Implementar persistência com base em JPA.
- Implementar regras de negócio na plataforma JEE, através de EJBs.
- Implementar sistema cadastral Web com base em Servlets e JSPs.
- Utilizar a biblioteca Bootstrap para melhoria do design.
- No final do exercício, o aluno terá criado todos os elementos necessários para exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para lidar com contextos reais de aplicação.

Códigos solicitados estão no repositório:

<https://github.com/luizfabriciorex/UNESA/tree/main/terceiroPeriodo/pratica-4>

#### Resultado da execução dos códigos:

O 3º procedimento utiliza a biblioteca do framework Bootstrap, o que torna a interface gráfica muito mais bonita e agradável.

A fig. 9 mostra a mesma listagem de produtos do procedimento anterior, mas com uso de Bootstrap, conforme solicitado no enunciado da Missão.

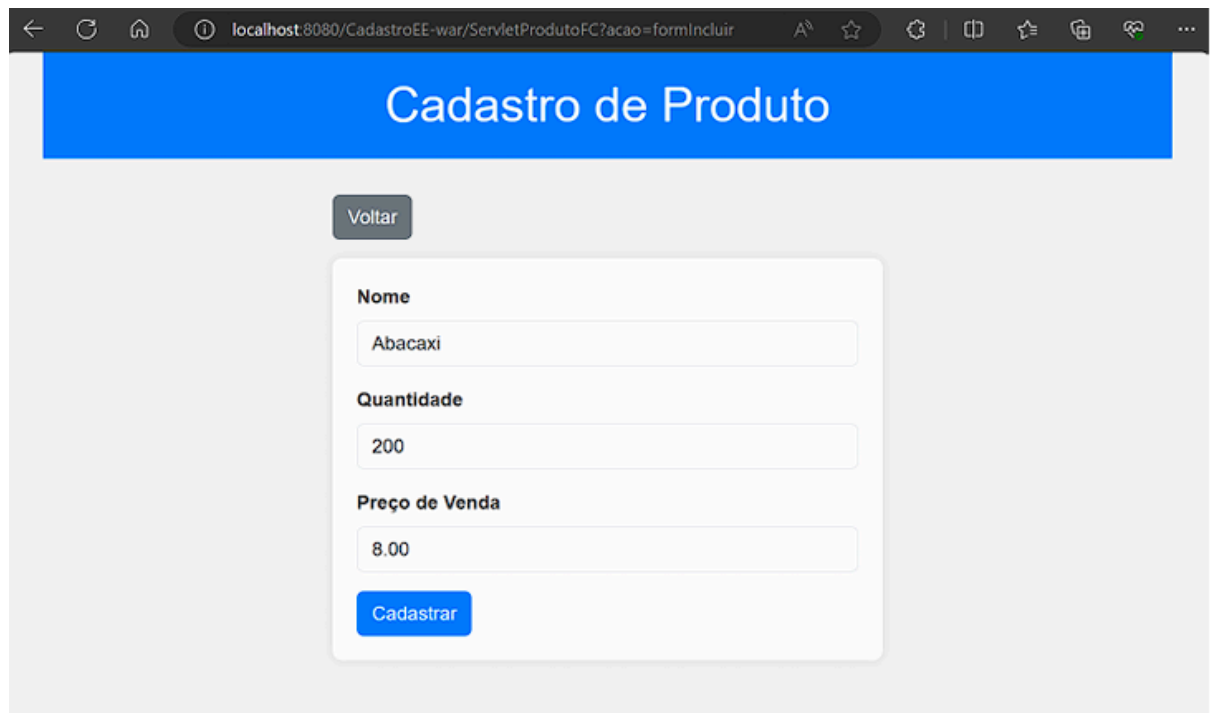


The screenshot shows a web browser window with the URL `localhost:8080/CadastroEE-war/ServletProdutoFC?acao=listar`. The page has a blue header with the title "Listagem de Produtos". Below the header is a table with the following data:

ID	Produto	Quantidade	Preço	Ações
1	Banana	100	R\$ 5,00	<button>Alterar</button> <button>Excluir</button>
2	Laranja	500	R\$ 2,00	<button>Alterar</button> <button>Excluir</button>
3	Manga	800	R\$ 4,00	<button>Alterar</button> <button>Excluir</button>
4	Tangerina	600	R\$ 7,00	<button>Alterar</button> <button>Excluir</button>

Below the table is a blue button labeled "Cadastrar Produto".

**Figura 9.** ServletProdutoFC em execução, com uso de Bootstrap



Cadastro de Produto

Voltar

Nome  
Abacaxi

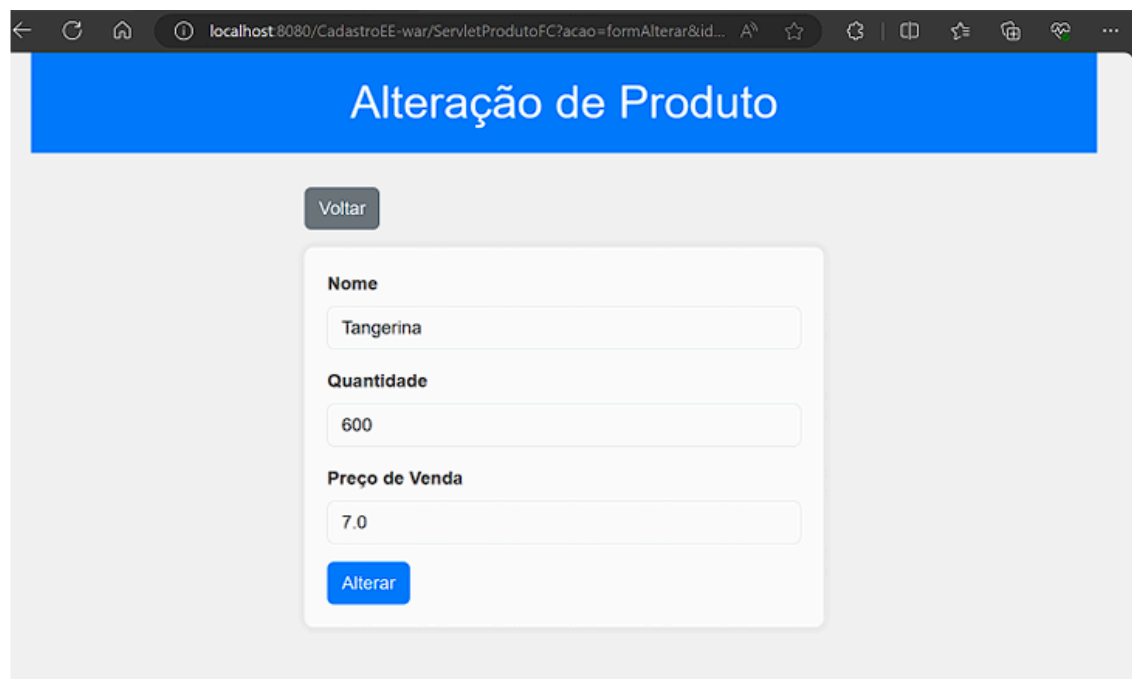
Quantidade  
200

Preço de Venda  
8.00

Cadastrar

**Figura 10.** Tela de Cadastro de Produto

Na fig. 10 é mostrada a tela de cadastro de produto, com uso de Bootstrap. Assim como na fig. 11 é mostrada a tela de alteração dos dados de produto cadastrado.



Alteração de Produto

Voltar

Nome  
Tangerina

Quantidade  
600

Preço de Venda  
7.0

Alterar

**Figura 11.** Tela de Alteração de Produto já cadastrado.

A) Como o framework Bootstrap é utilizado?

O framework Bootstrap é amplamente utilizado para desenvolver interfaces de usuário responsivas, mobile first, para websites e aplicações webs. Oferece um conjunto robusto de ferramentas baseadas em HTML, CSS e JavaScript, que incluem modelos pré-desenvolvidos para botões, formulários, navegação e outros elementos de interface, além de um sistema de grid flexível para layout. Isso permite aos desenvolvedores rapidamente construir sites esteticamente agradáveis, funcionais, e se adaptem automaticamente a diferentes tamanhos de tela e dispositivos, sem a necessidade de escrever código de estilização CSS desde o início.

B) Por que o Bootstrap garante a independência estrutural do HTML?

O Bootstrap assegura a independência estrutural do HTML, ao fornecer um conjunto de classes CSS pré-definidas, assim como componentes de interface que podem ser facilmente integrados ao HTML. Em razão disso, ao invés de escrever e ajustar diferentes estilos CSS personalizados para cada elemento, os desenvolvedores podem simplesmente usar as classes do Bootstrap para alcançar um design consistente e responsivo.

Portanto, a estrutura do HTML permanece limpa e desacoplada de estilos específicos, o que facilita a manutenção e a escalabilidade do código.

C) Qual a relação entre o Bootstrap e a responsividade da página?

O Bootstrap é um framework de desenvolvimento web que facilita a criação de páginas responsivas. Oferece um sistema de grid flexível, componentes pré-desenvolvidos e classes CSS que se ajustam automaticamente ao tamanho da tela do dispositivo, de maneira a garantir que o layout da página seja otimizado para desktops, tablets e smartphones. Isso simplifica o processo de design responsivo, o que permite que desenvolvedores criem sites que proporcionam uma experiência de usuário consistente em diversos dispositivos com menos esforço.