

Lista 29 – Back-end Spring Boot e Front-End Thymeleaf

O objetivo desta lista é praticar os conceitos vistos nas aulas dos dias 09.03, 10.03 e 16.03 e implementar o *back-end* em Spring Boot, incluindo as validações e o *front-end* em Thymeleaf com o CRUD, incluindo as mensagens de retorno ao usuário. O sistema ficará semelhante ao mostrado abaixo, mas voltado ao gerenciamento de funcionários.

The screenshots show a web application for book management. The interface includes a sidebar with navigation links: Home, Livros, + Cadastrar, and Listar. The main content area displays the application's functionality.

Screenshot 1: Home Page
The Home page features a header with the application name "DEV-TI" and a sidebar. The main content area displays a "Apoio" section with logos for Fapesc, Governo de Santa Catarina, UNOESC, and Spring Framework. The footer indicates "© 2023 DEV-TI. Todos os direitos reservados."

Screenshot 2: List of Books
The "Lista de Livros" page shows a search bar with the placeholder "Entre com o título" and a "Buscar" button. A dropdown menu for "Registros por página" is set to 4, and a "Limpar" button is present. The table below lists books with columns for Id, Título, # Páginas, Autor, and Ação.

Id	Título	# Páginas	Autor	Ação
16	Em busca dos anéis perdidos	100	Fulano da Silva	[Edit] [Delete]
1	O hobbit 1	1	J.R.R.Tolkien	[Edit] [Delete]
10	O hobbit 10	10	J.R.R.Tolkien	[Edit] [Delete]
11	O hobbit 11	11	J.R.R.Tolkien	[Edit] [Delete]

Below the table, it states "Total de registros: 16" and provides pagination controls: "Anterior", "1", "2", "3", "4", "Próximo". The footer shows "Página 1 de 4" and "© 2023 DEV-TI. Todos os direitos reservados."

Screenshot 3: Create Book Form
The "Formulário de Cadastro de Livros" page includes a "Listar Livros" button. A red message box indicates validation rules: "É obrigatório informar um título", "O livro deve ter pelo menos 1 página", and "O nome do autor deve ter entre 3 e 100 caracteres". The form fields are: "Título" (empty), "# de páginas" (0), and "Autor" (zê). "Incluir" and "Cancelar" buttons are at the bottom. The footer shows "© 2023 DEV-TI. Todos os direitos reservados."

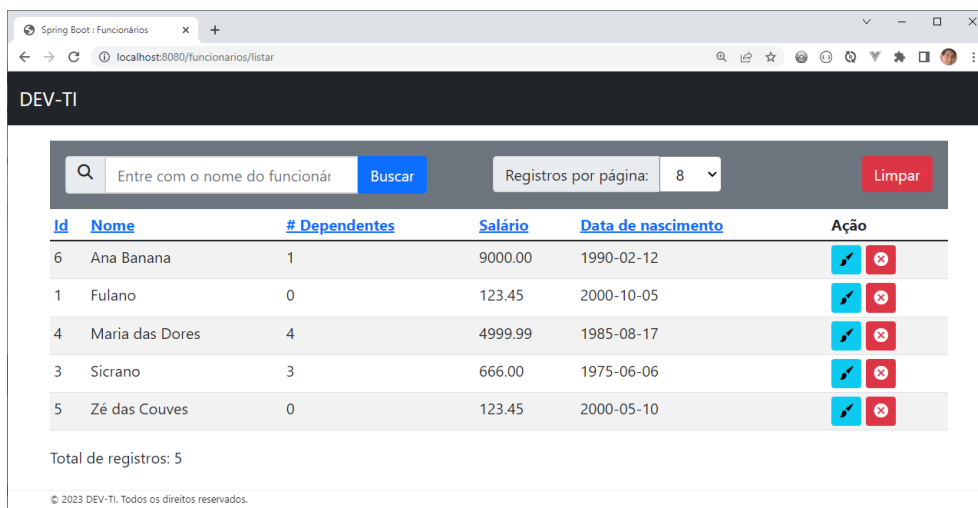
Screenshot 4: List of Books after Update
The "Lista de Livros" page shows a green message box: "Livro alterado com sucesso". The search bar and pagination controls are updated. The table now shows 16 records, with the first row being "Em busca dos anéis perdidos" by Fulano da Silva. The footer shows "© 2023 DEV-TI. Todos os direitos reservados."











Como base baixe e importe este projeto: [sistema_base_lista28.zip](#), que implementa as funcionalidades da lista 28.

Abaixo alguns exemplos implementados pela API REST do projeto da lista 28:

- <http://localhost:8080/api/funcionarios>
- <http://localhost:8080/api/funcionarios/1>
- <http://localhost:8080/api/funcionarios/xml/1>
- <http://localhost:8080/api/funcionarios/buscar?nome=ano>
- <http://localhost:8080/api/funcionarios/listar-paginas?size=2>
- <http://localhost:8080/api/funcionarios/paginas?nome=an&tamPagina=3&ordenacao=nome>
- <http://localhost:8080/api/funcionarios/faixa-salarial?minimo=500&maximo=5000>
- <http://localhost:8080/api/funcionarios/dependentes>

Já o end-point <http://localhost:8080/funcionarios/listar> do projeto da lista 28 abre a seguinte página:



Id	Nome	# Dependentes	Salário	Data de nascimento	Ação
6	Ana Banana	1	9000.00	1990-02-12	 
1	Fulano	0	123.45	2000-10-05	 
4	Maria das Dores	4	4999.99	1985-08-17	 
3	Sicrano	3	666.00	1975-06-06	 
5	Zé das Couves	0	123.45	2000-05-10	 

Total de registros: 5

© 2023 DEV-TI. Todos os direitos reservados.

Para modificar o projeto da lista 28 de forma a implementar as funcionalidades desejadas, siga o roteiro abaixo e use como base a apresentação 'aula 50b - projeto básico spring (09.02.2023).pptx' e também o sistema de CRUD de livros desenvolvido durante as últimas aulas.

1. Adicione o suporte para *layouts* do Thymeleaf (*slide 127*).
2. Modifique a estrutura de pastas e arquivos do projeto tomando por base o *slide 128*. O arquivo *paginacao.html* deverá ser renomeado para *lista.html* e colocado dentro da pasta *funcionario*. As imagens podem ser baixadas neste [link](#).
3. O conteúdo do arquivo *style.css* está nos *slides 130 e 131*.
4. O conteúdo do arquivo *layout.html* está no *slide 132*.
5. Os fragmentos *header.html*, *footer.html*, *sidebar.html* e *home.html* estão nos *slides 133 a 136*.
6. Adapte o arquivo *lista.html* de forma a usar fragmentos. Remova também o código desnecessário da seção `<head>`. Acrescente, por fim, o código JavaScript. O modelo pode ser visto nos *slides 138 a 140*.
7. Acrescente o controlador `HomeController` (*slide 141*).
8. Na classe `FuncionarioController` acrescente os métodos do *slide 142*, adaptando-os conforme necessário.
9. Crie o arquivo *cadastro.html* na pasta *funcionario* dentro de *templates*. Adapte o código do *slide 144*.
10. O código para o arquivo *modal.html* se encontra no *slide 145*.
11. Para implementar as mensagens de *feedback* ao usuário, use o código do *slide 147* para o arquivo *alert.html*.
12. Acrescente, no arquivo *lista.html*, a instrução que carrega o fragmento criado acima (*slide 148*).
13. No arquivo `FuncionarioController`, acrescente as linhas que criam o atributo `sucesso` com suas respectivas mensagens de retorno, assim como o `RedirectAttributes` e `addFlashAttribute`. (*slide 149*).
14. No arquivo *layout.html* acrescente o código JavaScript que remove automaticamente a mensagem de alerta após um determinado período de tempo (*slide 150*).
15. Na classe `Funcionario`, adicione, à sua escolha, restrições de tamanho de campo, unicidade e que não permitam valores nulos nos campos (*slide 153*).
16. Use o *slide 155* como modelo para acrescentar anotações de validação na classe `FuncionarioDTO`.
17. Acrescente o método `fromDTO()` na *interface* `FuncionarioService` e na classe `FuncionarioServiceImpl` (*slide 156*).
18. Altere a classe `FuncionarioController` de forma a usar a classe `FuncionarioDTO` e também implementar a validação de dados com as anotações e métodos `@Valid`, `BindingResult`, `hasErrors()` (*slides 157 e 158*).

19. O código para o fragmento *validacao.html* está no *slide* 160.
20. Modifique o arquivo *cadastro.html* a fim de usar o objeto DTO. Inclua também o fragmento de validação (*slide* 161).
21. Use a apresentação 'aula 67 - documentação com o swagger (17.03.2023).pptx' e o sistema [funcionarios.zip](#) como base e implemente a documentação através do Swagger.
 - Teste se a documentação está abrindo no endereço <http://localhost:8080/swagger-ui/index.html>.
 - Faça testes da API REST com o Swagger (*slide* 16).
 - Siga os *slides* 18 a 21 para importar para o Postman as definições da API REST do sistema. Use os métodos da API recém importada para fazer alguma requisição ao *back-end*.