



EST – IPCB

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Aplicações Distribuídas

A minha primeira Aplicação Spring Boot Web MVC com Templates

Thymeleaf

3º Ano / 1º Semestre – 2022/2023

Atividade Prática n.º2 - A Minha Primeira Aplicação Spring Boot Web MVC + Thymeleaf + Bootstrap

De seguida, pretende-se que desenvolva a sua primeira aplicação Spring Boot Web baseada num Controlador MVC. Adicionalmente será utilizada a biblioteca de Templates Thymeleaf e o front-end Bootstrap.

Ao longo desta atividade, pretende-se que ao realizar os passos seguintes experimente diversas funcionalidades do Spring Boot e possa ficar a conhecer algumas características das aplicações Spring Boot MVC Web.

Mais à frente, numa das próximas atividades será aprofundado o estudo/uso do Spring Boot MVC e do Thymeleaf.

Para melhor compreensão dos conteúdos deste guião sugere-se a consulta do bloco de slides: AD-2023-Bloco_1_Spring_e_Spring_Boot.pdf.

Passo 1: Criar uma Aplicação DemoMVC-Thymeleaf

Utilize o Spring Initializr (<https://start.spring.io>) para adicionar ao ficheiro pom.xml os Starter Spring Web, e Thymeleaf.

1. Escolha o **Project** do tipo **Maven Project**.
2. Escolha a linguagem **Java**.
3. Escolha a versão do Spring Boot. Utilize a última versão estável disponível.
4. Introduza os metadados do projecto – group ID, artifact ID, name of the project, project description, e package name.
5. Escolha **Packaging** as **Jar**.
6. Escolha **Java 17** ou **Java 19** (a versão mínima é Java 8).
7. Adicione as dependências – **Spring Web** e **Thymeleaf**.
8. Clique o botão **Generate**.
9. Descomprima o ficheiro .zip.

Passo 2: Criar o modelo Utilizador e um controlador ControladorUtilizador

A classe Utilizador mantém os dados de um utilizador. O controlador atualiza o modelo Utilizador e atualiza as vistas (páginas Web).

```
public class Utilizador {
    private Integer id; //Gerado aleatoriamente pela
    aplicação.
    private String nome;
    private String email;
    private String password;
    //getters e setters
}

@Controller
public class ControladorUtilizador
{
    // Métodos que executam as ações
}
```

Passo 3: Implementar no ControladorUtilizador os seguintes métodos ação anotados com `@GetMapping("/")` e `@PostMapping("/registar")`, respetivamente:

```
//URI: /
String index();
```

Este método retorna a Vista **index.html** que contém um formulário Web para introdução dos dados do utilizador a registar.

```
//URI: /registar
ModelAndView registaUtilizador(@ModelAttribute Utilizador
utilizador);
```

Este método permite o registo de um utilizador.

Recebe os dados do utilizador submetidos no formulário Web da página **index.html**. O Id pode ser gerado aleatoriamente usando os serviços de um objeto do tipo **Random**.

Este método retorna os dados num modelo Utilizador e a Vista **dados-utilizador.html** num objeto **ModelAndView**.

Passo 4: Criar os ficheiros das Vistas na pasta resources/templates do projeto da aplicação MVC.

- index.html
- dados-utilizador.html

Descarregue do moodle os ficheiros html de base às Vistas.

Nota: Estes podem ser criados a partir do starter template Bootstrap 5 disponível em: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>

Passo 5: Inclua nos htmls das vistas as tags th: por forma a torná-los nas duas templates Thymeleaf.

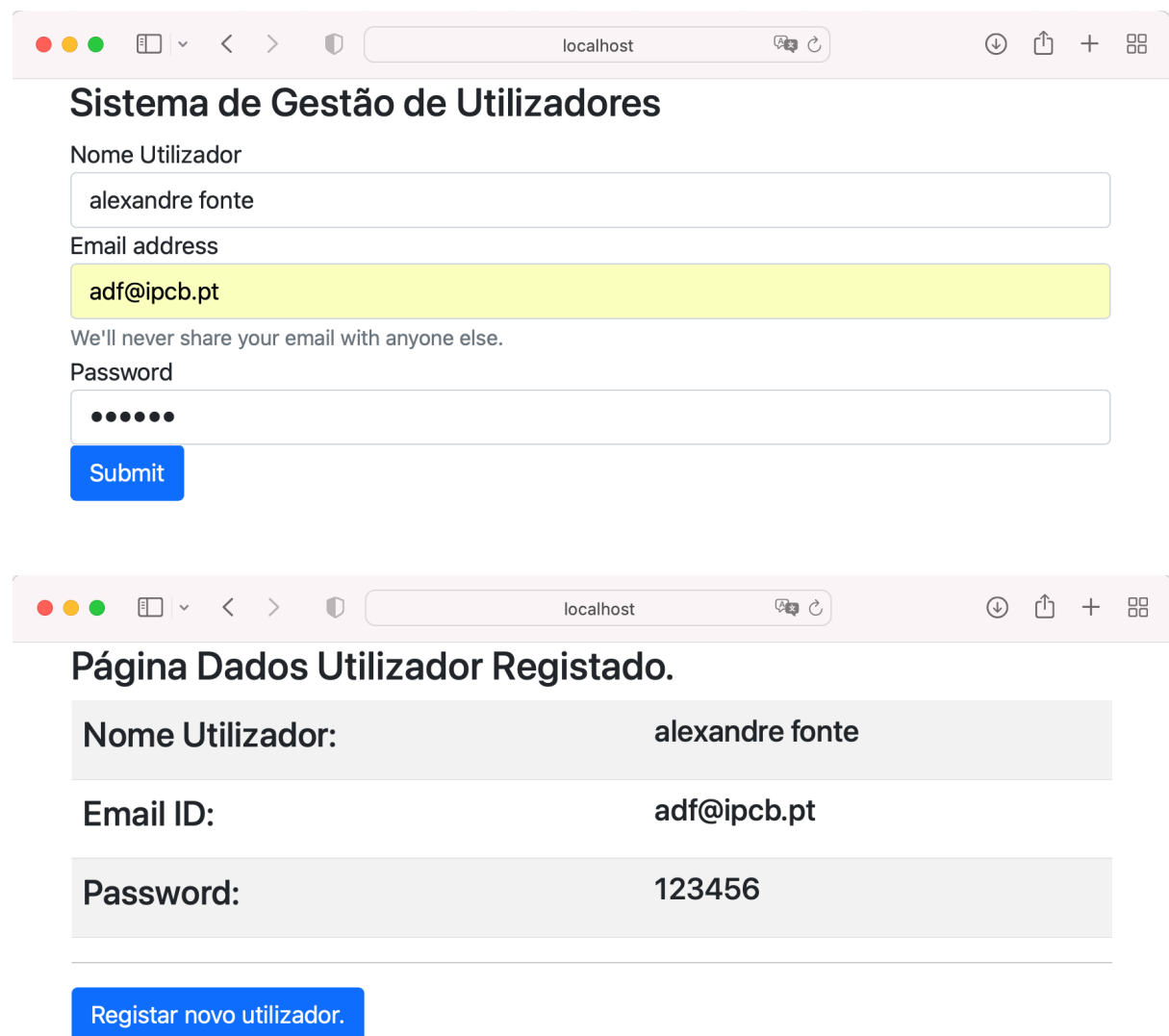
Comece por adicionar o atributo do namespace thymeleaf, na tag html de abertura:

```
<html lang="en" xmlns:th="https://thymeleaf.org">
```

Consulte os exemplos dos slides de apoio ou na página <https://www.baeldung.com/thymeleaf-in-spring-mvc>.

Passo 6: Teste a aplicação.

No final a aplicação terá *outputs* idênticos aos seguintes:



Sistema de Gestão de Utilizadores

Nome Utilizador

Email address

We'll never share your email with anyone else.

Password

Submit

Página Dados Utilizador Registado.

Nome Utilizador:	alexandre fonte
Email ID:	adf@ipcb.pt
Password:	123456

Registrar novo utilizador.

Passo 7: Desafios adicionais.

7.1. Adicione à aplicação a funcionalidade Listagem/Consulta de todos os utilizadores

- Cada utilizador registado é armazenado numa lista de utilizadores (ver ajuda).
- Inclua um novo método que permite listar todos os utilizadores:

//URI: /listar

```
@GetMapping("/listar")
```

```
public String getTodosUtilizadores(Model modelo)
```

Em vez de retornar um **ModelAndView** este método retorna o nome/string da vista **lista-utilizadores.html**.

O parâmetro **Model** **modelo** é instanciado pelo IoC container automaticamente. É preciso adicionar ao **Model** a lista de utilizadores como um atributo (usar o método **setAttribute()**).

Em vez de retornarmos um **ModelAndView** retornamos o nome da vista.

- Incluir na Vista Dados Utilizador Registado também a informação sobre o ID e um novo Botão Listar todos os utilizadores.
- Adaptar o ficheiro **lista-utilizadores.html** por forma a receber a lista e a tabela html mostrar os dados de cada utilizador linha a linha.

Ajudas:

- Criar uma classe de configuração que disponibiliza um método que retorna um @Bean do tipo **List<Utilizador>** com o nome **bdutilizadores**;
- Obtenha no controlador **ControladorUtilizador** o acesso a este @Bean (conforme fez na ficha n.º1) usando as anotações @Qualifier e @Autowired.
- A Thymeleaf permite incluir ciclos for usando o atributo **th:each**, por exemplo, a seguir mostra-se como podemos percorrer uma lista de **estudantes** e ecoar os dados de cada elemento **estudante**.

```
<tr th:each="estudante:${estudantes}">  
  <td th:text="${estudante.id}" />  
  <td th:text="${estudante.nome}" />  
</tr>
```

localhost

Página Dados Utilizador Registrado.

ID Utilizador:	45445
Nome Utilizador:	Vasco Soares
Email ID:	vasco.g.soares@ipcb.pt
Password:	12345678

Registrar novo utilizador.

Listar todos os utilizadores.

localhost

Listagem de Todos os Utilizadores Registrados

Nome	Email	Password
alexandre fonte	adf@ipcb.pt	123456
Vasco Soares	vasco.g.soares@ipcb.pt	1234567

Registrar novo utilizador.

7.2. Adicione à aplicação as funcionalidades de Remoção e Edição de um Utilizador.

- Inclua dois novos métodos que permitem respetivamente eliminar e editar um utilizador pelo seu Id anotados com `@GetMapping("/remover/{id}")` ou `@GetMapping("/editar/{id}")`, respetivamente:

//URI: /remover/{id}

```
String removerUtilizador(@PathVariable("id") Integer id,  
Model modelo)
```

Este método após remover o utilizador lista todos os utilizadores.

Ajuda: Percorra a lista e remova o utilizador usando um Iterator.

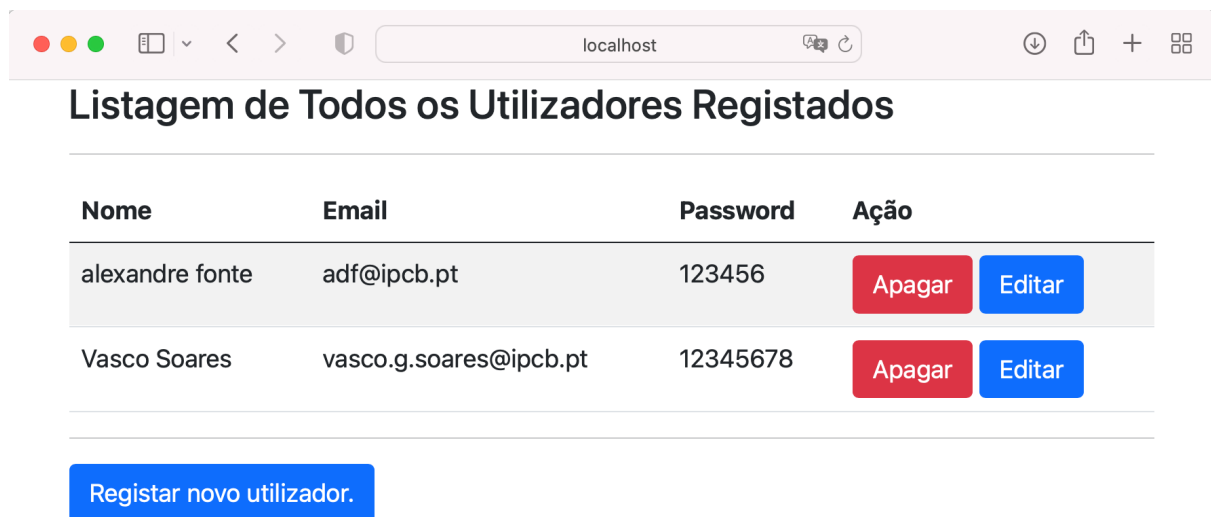
//URI: /editar/{id}

```
ModelAndView editarUtilizador(@PathVariable("id") Integer  
id)
```

Este método carrega a página **editar-dados.html** a qual corresponde basicamente à página de registo. Esta também submete para **/registar**.

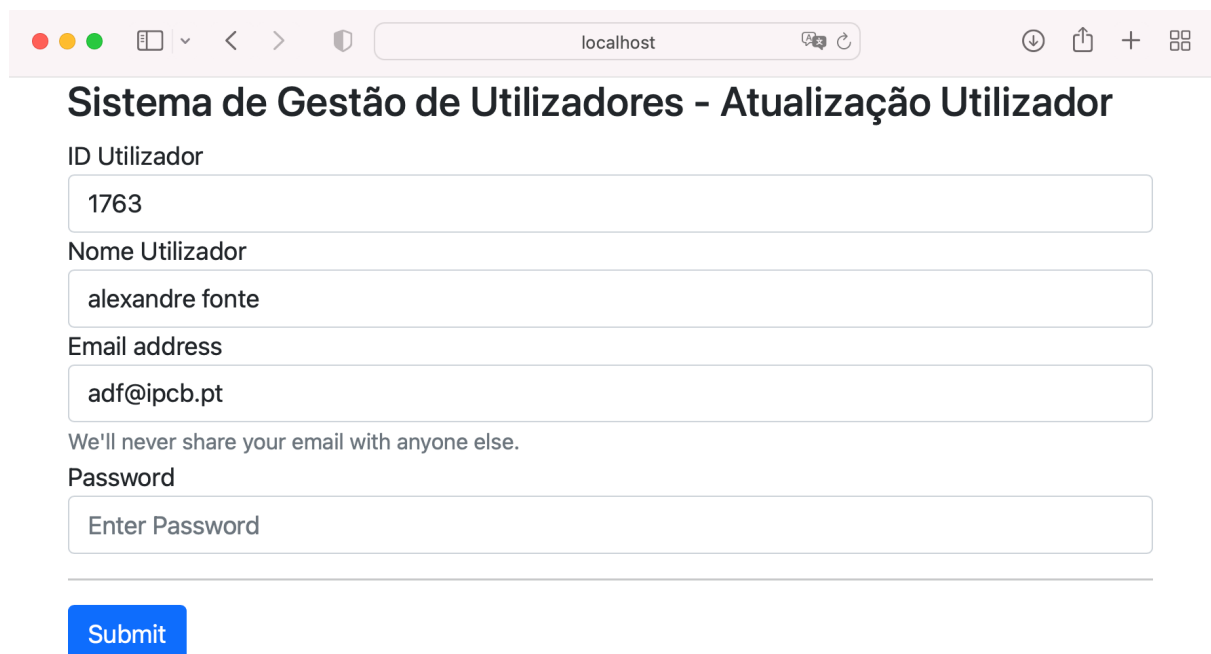
- Na vista **lista-utilizadores.html**, incluir na tabela dos Utilizadores Registados uma coluna Ação e dois Botões Apagar e Editar em cada Utilizador registado:

```
<td><a th:href="@{/remover/{id} (id=${utilizador.id})}"  
      class="btn btn-danger">Apagar</a>  
      <a th:href="@{/editar/{id} (id=${utilizador.id})}"  
      class="btn btn-primary">Editar</a>  
</td>
```



Nome	Email	Password	Ação
alexandre fonte	adf@ipcb.pt	123456	<button>Apagar</button> <button>Editar</button>
Vasco Soares	vasco.g.soares@ipcb.pt	12345678	<button>Apagar</button> <button>Editar</button>

Registar novo utilizador.



Sistema de Gestão de Utilizadores - Atualização Utilizador

ID Utilizador

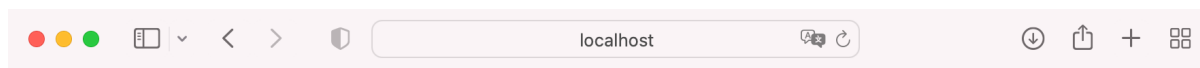
Nome Utilizador

Email address

We'll never share your email with anyone else.

Password

Submit



Listagem de Todos os Utilizadores Registados

Nome	Email	Password	Ação	
Vasco Soares	vasco.g.soares@ipcb.pt	12345678	Apagar	Editar
Professor alexandre fonte	adf@ipcb.pt	123456	Apagar	Editar

Registar novo utilizador.

FIM