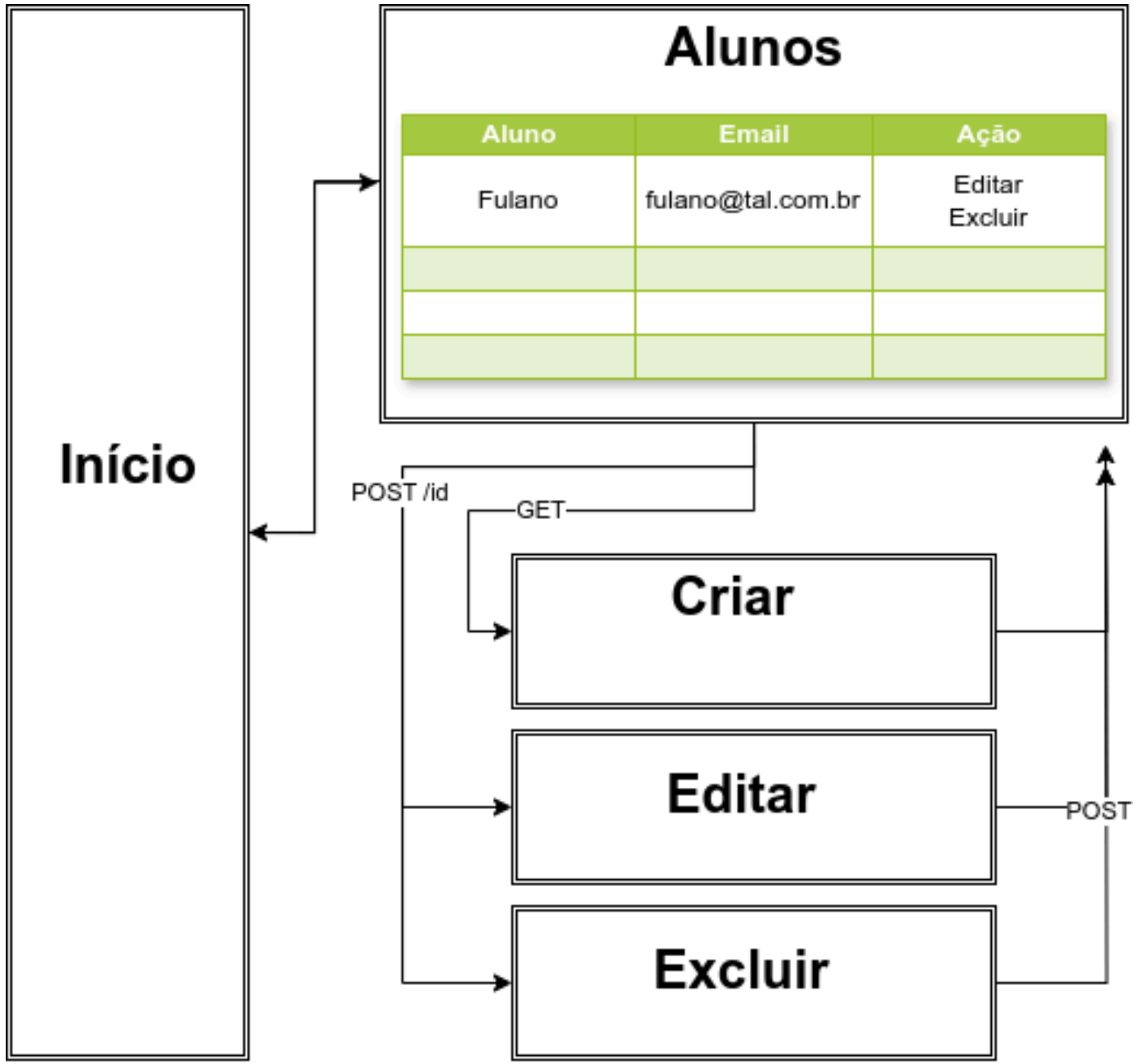
Roteiro

- Finalizar o CRUD
- Aplicar um framework CSS

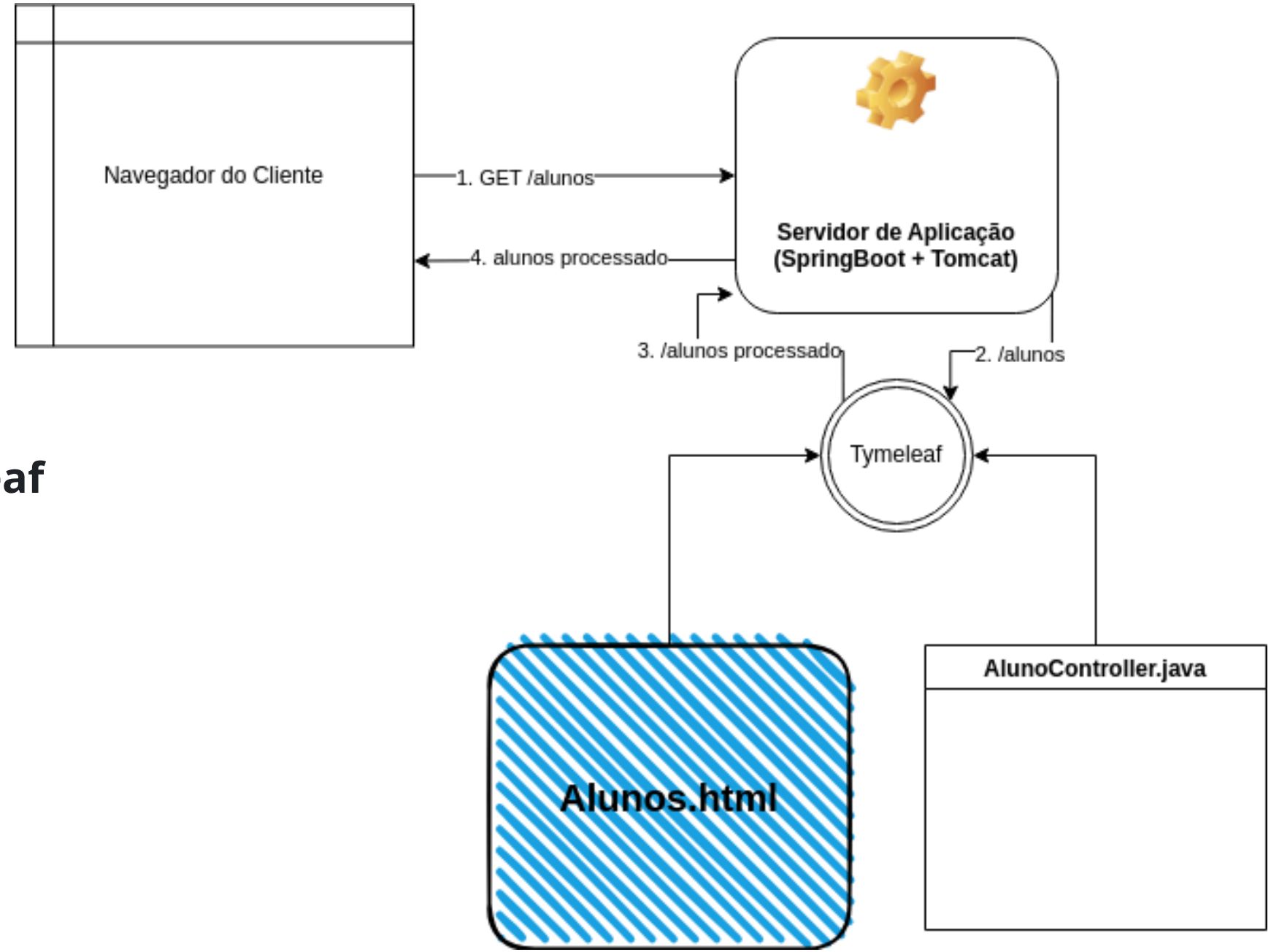
Nosso CRUD

- Ler: alunos.html
- Criar: aluno-create.html
- Remover: aluno-delete.html
- Editar: aluno-update.html

Navegação do CRUD:



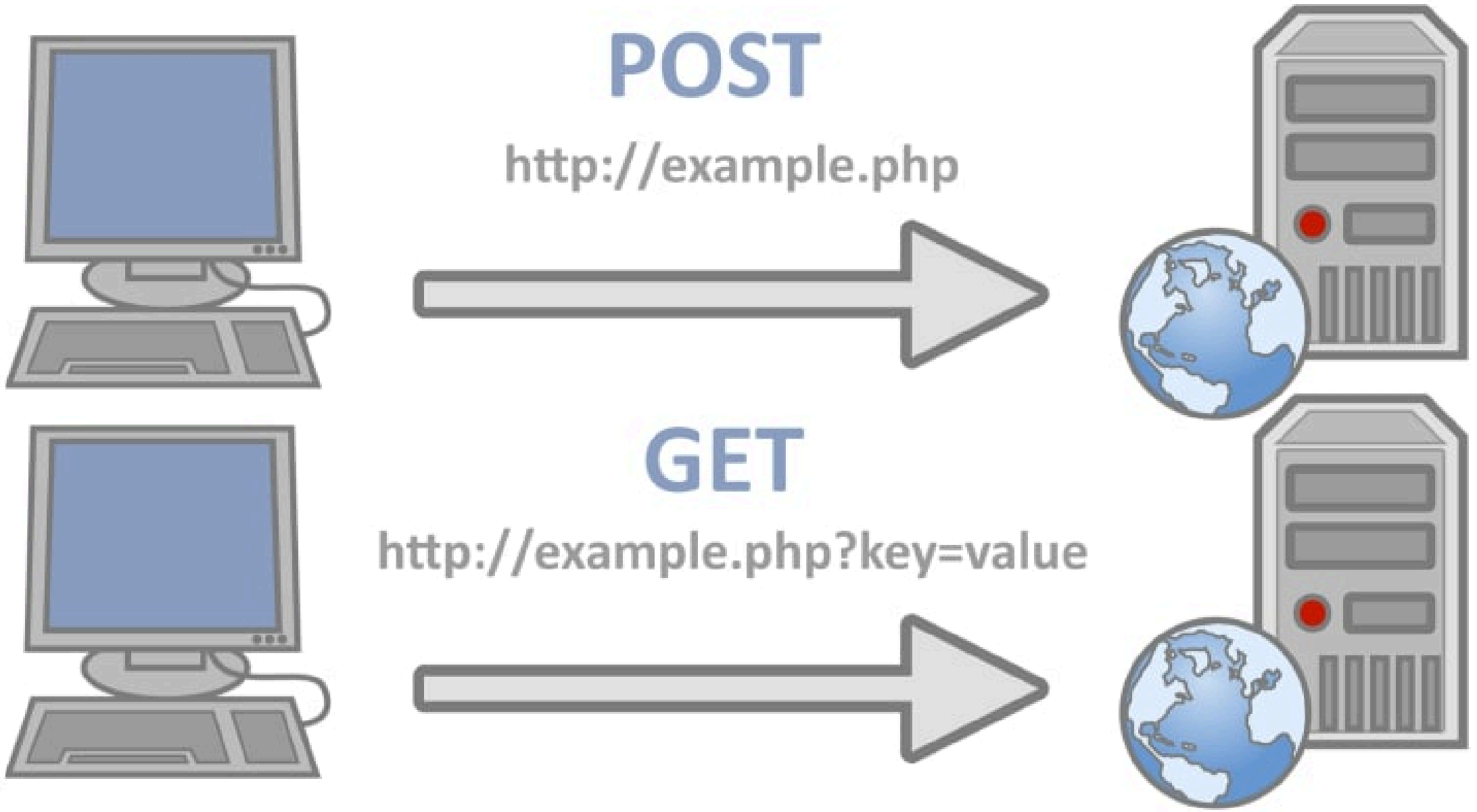
Thymeleaf


Navegador do Cliente
1. GET /alunos
4. alunos processado
Servidor de Aplicação
(SpringBoot + Tomcat)
3. /alunos processado
2. /alunos
Tymeleaf
Alunos.html
AlunoController.java

Get vs Post

- GET: Solicita dados de um recurso específico.
- POST: Submete dados para serem processados por um recurso.

É seguro utilizar o método GET para solicitar dados, mas não para enviar dados sensíveis.


POST
http://example.php
GET
http://example.php?key=value

Início: HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>UEG - Desenvolvimento Web 2025</title>
</head>
<body>
    <h1>Início</h1>
    <p>
        Bem-vindo ao curso de Desenvolvimento Web!
    </p>
    Cadastros:
    <ul>
        <li><a href="/">Inicio</a></li>
        <li><a href="/alunos">Alunos</a></li>
    </ul>
</body>
</html>
```

Início: Java - AlunoControoler

```java
@Controller
public class AlunoController {

  @GetMapping("/")
  public String index() {
    return "index";
  }
}
```

Alunos: Leitura - HTML

```
<h1>Lista de Alunos</h1>
<ul>
    <li><a href="/">Inicio</a></li>
    <li><a href="/alunos">Alunos</a></li>
    <li><a href="/alunos/create">Novo Aluno</a></li>
</ul>

<table class="table">
    <tr><th>Nome</th> <th>Matrícula</th><th>Açõe</th></tr>

    <tr th:each="aluno, iter: ${alunos}">
        <td th:text="${aluno.nome}"></td>
        <td th:text="${aluno.email}"></td>
        <td><ul>
                <li><a th:href="@{'/alunos/update/' + ${iter.index}}">Atualizar</a>
                </li>
                <li><a th:href="@{'/alunos/delete/' + ${iter.index}}">Excluir</a></li>
        </ul></td>
    </tr>
</table>
```

Alunos: Leitura - Java

```
@Controller
public class AlunoController {

    static List alunos = new ArrayList<>();

    static {
        alunos.add(Map.of("nome", "João", "email", "joao@localhost"));
        alunos.add(Map.of("nome", "Maria", "email", "maria@localhost"));
    }

    @GetMapping("/alunos")
    public String getHome(Model model) {
        model.addAttribute("alunos", alunos);
        return "alunos";
    }
...
}
```

Aluno: Criar - HTML

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>UEG - Desenvolvimento Web 2025: Aluno</title>
</head>

<body>
<h1>Cadastro de Aluno</h1>

<form action="/alunos/create" method="post">
    <label>Nome</label>
    <input type="text" name="nome">

    <label>Email</label>
    <input type="text" name="email">

    <button>Salvar</button>
</form>
</body>
</html>
```

Aluno: Criar - JAVA

```
@Controller
public class AlunoController {

    @GetMapping("/alunos/create")
    public String getCreate() {
        return "aluno-create";
    }

    @PostMapping("/alunos/create")
    public String postCreate(@RequestParam String nome, @RequestParam String email) {
        alunos.add(Map.of("nome", nome, "email", email));
        return "redirect:/alunos";
    }
...
}
```

Aluno: Editar - HTML

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>UEG - Desenvolvimento Web 2025: Aluno</title>
</head>

<body>
<h1>Cadastro de Aluno</h1>
<form action="/alunos/update" method="post">
    <input type="hidden" name="id" th:value="${id}">
    <label>Nome: </label>
    <input type="text" name="nome" th:value="${aluno.nome}">

    <label>Email</label>
    <input type="text" name="email" th:value="${aluno.email}">

    <button class="waves-effect waves-light btn">Salvar</button>
</form>
</body>
</html>
```

Alunos: Editar - JAVA

```java
@Controller
public class AlunoController {

    @GetMapping("/alunos/update/{id}")
    public String getUpdate(@PathVariable int id, Model model) {
        model.addAttribute("aluno", alunos.get(id));
        model.addAttribute("id", id);
        return "aluno-update";
    }

    @PostMapping("/alunos/update")
    public String postUpdate(@RequestParam int id, @RequestParam String nome, @RequestParam String email) {
        alunos.set(id, Map.of("nome", nome, "email", email));
        return "redirect:/alunos";
    }
...
}
```

Aluno: Excluir - HTML

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>UEG - Desenvolvimento Web 2025: Aluno</title>
</head>

<body>
<h1>Cadastro de Aluno</h1>
<form action="/alunos/delete" method="post">
    <input type="hidden" name="id" th:value="${id}">
    <label>Nome: </label>
    <input type="text" name="nome" disabled>

    <label>Email</label>
    <input type="text" name="email" disabled>

    <button>Excluir</button>
</form>
</body>
</html>
```

Alunos: Excluir - JAVA

```
@Controller
public class AlunoController {

    @GetMapping("/alunos/delete/{id}")
    public String getDelete(@PathVariable int id, Model model) {
        model.addAttribute("aluno", alunos.get(id));
        model.addAttribute("id", id);
        return "aluno-delete";
    }

    @PostMapping("/alunos/delete")
    public String postDelete(@RequestParam int id) {
        alunos.remove(id);
        return "redirect:/alunos";
    }
...
}
```

CRUD

Testar o equeleto do CRUD

- Cadastrar um aluno
- Editar um aluno
- Remover um aluno

APLICAR CSS

Frameworks de CSS

Pesquisar sobre os principais frameworks de CSS:

- Bootstrap: https://getbootstrap.com/
- Materialize: https://materializecss.com/
- Bulma: https://bulma.io/
- Foundation: https://get.foundation/

Escolher um para utilizar no projeto ao longo do curso, justificar a escolha.

O que é um framework de CSS?

- Um framework de CSS é um pacote de arquivos contendo um conjunto de estilos predefinidos.
- Facilita o desenvolvimento de sites e aplicações web.
- Acelera o desenvolvimento.
- Responsividade.
- Compatibilidade com navegadores.
- Documentação e suporte.

Escolha do framework

Escolha do framework: Materialize

Justificativa:

- Design moderno e limpo.
- Facilidade de uso.
- Sair da zona de conforto.

Paginas do nosso projeto

- index.html
- alunos.html
- aluno-create.html
- aluno-update.html
- aluno-delete.html

Aplicando um Framework CSS

Escolher um dos temas do Framework e tentar utiliza-lo:

- Contendo uma região de cabeçalho e uma de conteúdo
- Adicionar uma barra de navegação
- Adicione estilos aos componentes:
  - Tabela
  - Botões
  - Formulários
  - Mensagem

Proxima aula:

- Reutilização de código: Thymelaf Fragment

Materialize

Instalação via CDN: https://materializecss.com/getting-started.html

adicionar no `<head>` do arquivo HTML:

```
<!-- Compiled and minified CSS -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">

    <!-- Compiled and minified JavaScript -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
```