Aula 5

Desenvolvimento Web – Back End

Profª Luciane Yanase Hirabara Kanashiro

1

Conversa Inicial

2

Testes de software

- **Teste de software**
- **Testes funcionais**
- **Mensagens ao usuário**
- **Validação back end e front end**
- **Testes unitários com JUnit**

3

Teste de software

4

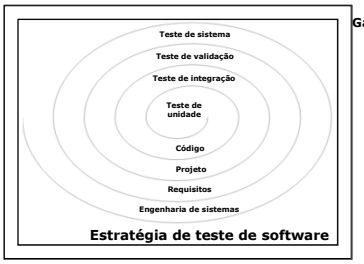
Testes de software

- **Importância:**
  - **Essenciais ao longo do ciclo de vida**
  - **Realizados em várias fases**
- **Objetivo:**
  - **Assegurar funcionamento do software**
  - **Garantir atendimento às especificações**
  - **Verificação e Validação (V&V)**

5

Custos X tempo

6



Estratégia de teste de software

Garantia de qualidade de software

Envolve diversos testes

Visam garantir a qualidade do software

Fonte: Pressman, 2021.

7

Teste de sistema

Compõe as atividades que verificam se todos os requisitos estão de acordo com o especificado e se foram realmente atendidos

Teste de validação

Mostra se o software atende aos requisitos, mostra se um programa faz o que é proposto a fazer

Teste de integração

Verifica se as unidades se comunicam, se integram corretamente sem falhas

Teste de unidade

Testar pequenas unidades em um sistema

8

Testes funcionais com Postman

9

Testes funcionais

- **Avaliar: software atende especificações funcionais**
- **Objetivo:**
  - **Verificar: funções operam conforme esperado**
  - **Garantir: cumprimento dos requisitos definidos**
- **Abrangência: testes unitários e de integração**

10

Principais características

- **Verificação das funcionalidades**
- **Verificação de conformidade com os requisitos**
- **Exploração de cenários de uso**
- **Verificação de entrada e saída**

11

Postman

- **Plataforma de colaboração para desenvolvimento de APIs**
- **Ferramentas:**
  - **Facilitam a criação**
  - **Teste**
  - **Documentação**
  - **Compartilhamento**

POSTMAN

12

Testes funcionais com o Postman

- Criação de solicitações HTTP
- Execução de testes automatizados
- Validação de respostas
- Criação de coleções de testes
- Integração com ambientes de desenvolvimento
- Relatórios e monitoramento

13

Mensagens ao usuário com Thymeleaf

14

Mensagens ao usuário

- Categorizadas:
  - Erros
  - Avisos
  - Informações

} moldam significativamente a experiência do usuário

15

Mensagens de erro

- Impacto na experiência do usuário:
  - Frustrações e desistência da aplicação
  - Mensagem confusa ou genérica
- Importância da clareza: informações precisas e sugerir soluções tangíveis

16

Mensagens de aviso

- Informações orientam sobre ações ou eventos iminentes
- Forma preventiva:
  - Alertas possíveis problemas
  - Fornecem instruções

17

Mensagens informativas

- Contexto valioso aos usuários
- Detalhes sobre ações concluídas com sucesso
- Atualizações do sistema
- Informações relevantes

18

Mensagens

- Thymeleaf para mensagens ao usuário
- Mensagens
  - Simples
  - Dinâmicas
  - Erro
  - Condicionalmente estilizadas

19

Mensagens

- Exibição simples:

```
<div th:if="${authenticated}">
    <p>Bem-vindo, usuário!</p>
</div>
```

- Exibição de mensagens dinâmicas

```
<p th:text="#{welcome.message}">Welcome!</p>
```

20

Mensagens

- Exibição de mensagens de erro

```
<div th:if="${#fields.hasErrors('campo')}">
    <p th:errors="*{campo}">Erro no
campo.</p>
</div>
```

- Mensagens condicionalmente estilizadas

```
<p th:text="${mensagem}" th:class="${sucesso} ? 'sucesso':
'erro'"></p>
```

21

Validação back end e front end

22

Relação entre testes funcionais e validação

- Garantir que o software atenda aos requisitos e funcione conforme esperado
- Papel dos testes funcionais na validação:
  - Parte prática da validação
  - Verificação: funcionalidades em conformidade com requisitos

23

Razões

- ☐ Validação de campos
- ☐ Garantir integridade, segurança e usabilidade
- ☐ Garantir formato e tipo correto
- ☐ Prevenir injeção de código malicioso

Contribuições

- ☐ Robustez e confiabilidade
- ☐ Estabelecer integridade dos dados
- ☐ Aumentar segurança contra ataques
- ☐ Melhorar experiência do usuário

24

Validação front end: Javascript

Abrir no codepen: https://codepen.io/pen/

25

Validação front end: HTML5

- Realizadas a partir dos atributos da tag input: type, pattern e required

```
<form>
  <h2>Validação de e-mail</h2>
  <input type="email" value="" placeholder="nome@email.com" required>
  <input type="submit" value="Enviar">
</form>
```

```
<form>
  <h2>Validação de fone</h2>
 <input type="tel" required="required" maxlength="15" name="telefone"
pattern="[0-9]{2}-[0-9]{4,6}-[0-9]{3,4}$" />
<input type="submit" value="Enviar">
</form>
```

26

Validação back end – Spring validation

- Parte crucial Spring
- Base na especificação de validação de objetos Java – Bean Validation

```
public class Usuario {
    @NotNull
    @Size(min = 3, max = 50)
    private String nome;

    @Email
    private String email;
    // Getters e Setters
}
```

27

Validação back end – Spring validation

- Dependência inclui bibliotecas para integrar o Bean Validation ao contexto do Spring

```
<dependency>
<groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

28

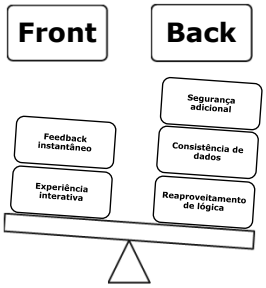
Validação back end – Spring validation

- Exemplo de controlador utilizando validação

```
@RestController
@RequestMapping("/usuarios")
public class UsuarioController {
    @PostMapping
    public ResponseEntity<String> cadastrarUsuario(@Valid @RequestBody
Usuario usuario) {
        // Lógica para processar o usuário
        return ResponseEntity.ok("Usuário cadastrado com sucesso!");
    }
}
```

29

Vantagens



30

Testes unitários com JUnit

31

JUnit

- **Framework testes unitários em Java**
- **Sintaxe clara e direta**
- **Testes facilmente compreensíveis**
- **anotação @Test**

32

JUnit exemplo

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class MinhaClasseTest {

    @Test
    public void testarMetodo() {
        MinhaClasse minhaInstancia = new MinhaClasse();
        assertEquals(6, minhaInstancia.multiplicar(2, 3));
    }
}
```

33

JUnit

- **Objetivo: garantir que cada método funcione conforme esperado**
- **Característica marcante: execução automatizada dos testes**

34

- **Esquema de cores semelhante a semáforo**
  - **Verde: sucesso**
  - **Azul: falha**
  - **Vermelho: erro**

35