Aluno: Lucas de Souza Ferreira

Matéria: Projeto de Bloco: Engenharia Disciplinada de Software

TP3

Teste de Performance 3 (TP3) - Relatório de Evidências

Visão Geral

Este relatório documenta a verificação dos requisitos do TP3 (Novo Sistema CRUD + Fuzzing) com base na tentativa de execução dos testes.

1. Verificação dos Requisitos

1.1 Sistema CRUD com Interface Web

- Requisito: Novo sistema robusto e eficiente.
- Status de Verificação: ✅ Compila e Executa (Firefox Validado).

1.2 Casos de Teste Avançados

- Requisito: Testes cobrindo falhas, timeouts e Fuzz Testing.
- Status:
- Unitários: ✅ Passaram com sucesso.
- Integração: ⚠️ Executados. Identificaram um Bug Lógico (Status 302 Found vs 200 OK no fluxo de erro). Isso é uma evidência positiva da efetividade do teste.
- E2E (Selenium): ✅ Executados com Sucesso (Firefox Headless).

2. Evidências de Código (Para o PDF)

Recomenda-se incluir trechos ou o conteúdo completo dos seguintes arquivos para demonstrar a implementação:

- Lógica de Negócio e Validação:
- src/main/java/com/cliente/projeto/crudpb/service/EventoService.java

```
@Service
public class EventoService {

    @Autowired
    private EventoRepository eventoRepository;

    public List<Evento> listarTodos() {
        return eventoRepository.findAll();
    }

    public Evento buscarPorId(Long id) {
        return eventoRepository.findById(id)
                .orElseThrow(() -> new RecursoNaoEncontradoException("Evento não
```

```
encontrado com ID: " + id));
    }

    public Evento criarEvento(Evento evento) {
        //  Regra de negócio (não pode ter nome duplicado)
        Optional<Evento> eventoExistente =
eventoRepository.findByNome(evento.getNome());
        if (eventoExistente.isPresent()) {
            throw new ValidacaoException("Já existe um evento com o nome: " +
evento.getNome());
        }
        return eventoRepository.save(evento);
    }

    public Evento atualizarEvento(Long id, Evento eventoAtualizado) {
        //Busca o evento (ou falha com 404)
        Evento eventoExistente = buscarPorId(id);

        //  Validação de nome duplicado (ignorando o próprio ID)
        Optional<Evento> conflito =
eventoRepository.findByNome(eventoAtualizado.getNome());
        if (conflito.isPresent() && !conflito.get().getId().equals(id)) {
            throw new ValidacaoException("O nome '" + eventoAtualizado.getNome() +
"' já está em uso por outro evento.");
        }

        //  Atualiza os dados
        eventoExistente.setNome(eventoAtualizado.getNome());
        eventoExistente.setDescricao(eventoAtualizado.getDescricao());

        return eventoRepository.save(eventoExistente);
    }

    public void deletarEvento(Long id) {
        // Verifica se existe antes de deletar (ou falha com 404)
        Evento eventoParaDeletar = buscarPorId(id);
        eventoRepository.delete(eventoParaDeletar);
    }
}
```

- Interface Web (Controller):
- src/main/java/com/cliente/projeto/crudpb/controller/EventoController.java

```
@Controller
@RequestMapping("/eventos")
public class EventoController {

    @Autowired
    private EventoService eventoService;

    // READ (Listagem) - Mapeia para /eventos
    @GetMapping
    public String listarEventos(Model model) {
        model.addAttribute("eventos", eventoService.listarTodos());
        return "lista-eventos"; // Renderiza o arquivo 'lista-eventos.html'
    }

    // CREATE (Mostrar formulário) - Mapeia para /eventos/novo
    @GetMapping("/novo")
    public String mostrarFormularioNovo(Model model) {
        model.addAttribute("eventoDTO", new EventoDTO("", ""));
        model.addAttribute("pageTitle", "Novo Evento");
        return "form-evento"; // Renderiza o arquivo 'form-evento.html'
    }

    // CREATE (Salvar) - Mapeia para POST /eventos
    @PostMapping
    public String salvarEvento(@Valid @ModelAttribute("eventoDTO") EventoDTO
```

```
eventoDTO,
                                BindingResult bindingResult, // Pega erros de
validação do DTO
                                Model model,
                                RedirectAttributes redirectAttributes) {

        if (bindingResult.hasErrors()) {
            model.addAttribute("pageTitle", "Novo Evento");
            return "form-evento";
        }

        try {
            // FAIL EARLY: Validação de regra de negócio (ex: nome duplicado)
            eventoService.criarEvento(eventoDTO.toEntity());

            redirectAttributes.addFlashAttribute("mensagemSucesso", "Evento criado
com sucesso!");
            return "redirect:/eventos";

        } catch (ValidacaoException ex) {
            model.addAttribute("pageTitle", "Novo Evento");
            model.addAttribute("mensagemErro", ex.getMessage());
            return "form-evento";
        }
    }

    // UPDATE (Mostrar formulário de edição) - Mapeia para /eventos/editar/{id}
    @GetMapping("/editar/{id}")
    public String mostrarFormularioEditar(@PathVariable Long id, Model model) {
        Evento evento = eventoService.buscarPorId(id); // Já lida com 404
        EventoDTO dto = new EventoDTO(evento.getNome(), evento.getDescricao());

        model.addAttribute("eventoDTO", dto);
        model.addAttribute("eventoId", id);
        model.addAttribute("pageTitle", "Editar Evento");
        return "form-evento";
    }

    // UPDATE (Atualizar) - Mapeia para POST /eventos/{id}
    @PostMapping("/{id}")
    public String atualizarEvento(@PathVariable Long id,
                                  @Valid @ModelAttribute("eventoDTO") EventoDTO
eventoDTO,
                                  BindingResult bindingResult,
                                  Model model,
                                  RedirectAttributes redirectAttributes) {

        if (bindingResult.hasErrors()) {
            model.addAttribute("pageTitle", "Editar Evento");
            model.addAttribute("eventoId", id);
            return "form-evento";
        }

        try {
            eventoService.atualizarEvento(id, eventoDTO.toEntity());
            redirectAttributes.addFlashAttribute("mensagemSucesso", "Evento
atualizado com sucesso!");
            return "redirect:/eventos";

        } catch (ValidacaoException ex) {
            model.addAttribute("pageTitle", "Editar Evento");
            model.addAttribute("eventoId", id);
            model.addAttribute("mensagemErro", ex.getMessage());
            return "form-evento";
        }
    }

    // DELETE - Mapeia para /eventos/deletar/{id}
    @GetMapping("/deletar/{id}")
    public String deletarEvento(@PathVariable Long id, RedirectAttributes
redirectAttributes) {
```

```
        eventoService.deletarEvento(id);
        redirectAttributes.addFlashAttribute("mensagemSucesso", "Evento deletado
com sucesso!");
        return "redirect:/eventos";
    }
}
```

- Testes Unitários:
- src/test/java/com/cliente/projeto/crudpb/service/EventoServiceTest.java
-

```
@ExtendWith(MockitoExtension.class) // Habilita o Mockito
class EventoServiceTest {

    @Mock // Cria um "dublê" do repositório
    private EventoRepository eventoRepository;

    @InjectMocks // Injeta os mocks acima no serviço
    private EventoService eventoService;

    @Test
    void deveLancarExcecao_QuandoCriarEventoComNomeDuplicado() {
        // 1. Cenário (Given)
        // Simulando que o repositório encontrou um evento com este nome
        when(eventoRepository.findByNome("Evento
Repetido")).thenReturn(Optional.of(new Evento()));

        Evento eventoNovo = new Evento("Evento Repetido", "Descricao");

        // 2. Ação e Verificação (When & Then)
        // Verificamos se a exceção de validação de negócio é lançada
        assertThrows(ValidacaoException.class, () -> {
            eventoService.criarEvento(eventoNovo);
        });

        // 3. Verificação Extra
        // Garantir que o "fail early" funcionou e NUNCA tentamos salvar
        verify(eventoRepository, never()).save(any());
    }
}
```

- Testes E2E (Selenium):
- src/test/java/com/cliente/projeto/crudpb/e2e/pageobjects/CadastroEventoE2ETest.java

```java
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class CadastroEventoE2ETest {

    @LocalServerPort
    private int port;

    private WebDriver driver;
    private String baseUrl;
    private FormularioEventoPage formularioPage;

    @BeforeAll
    static void setupClass() {
        WebDriverManager.firefoxdriver().setup(); // Baixa o GeckoDriver
    }

    @BeforeEach
    void setupTest() {
        org.openqa.selenium.firefox.FirefoxOptions options = new
org.openqa.selenium.firefox.FirefoxOptions();
        options.addArguments("--headless");
        // options.addArguments("--no-sandbox"); // Firefox usually doesn't need
this as
        // critically as Chrome
        // options.addArguments("--disable-dev-shm-usage");
        driver = new org.openqa.selenium.firefox.FirefoxDriver(options);

        baseUrl = "http://localhost:" + port;
        formularioPage = new FormularioEventoPage(driver);
    }

    @AfterEach
    void teardown() {
        if (driver != null) {
            driver.quit();
        }
    }

    /**
     * Teste de Interface (Artefato 2)
     * Valida o feedback de erro seguro na UI (Artefato 4)
     */
    @Test
    void deveMostrarMensagemDeErro_QuandoSubmeterFormularioComNomeVazio() {
        driver.get(baseUrl + "/eventos/novo");

        formularioPage.preencherFormulario("", "Descrição válida"); // Nome vazio
        formularioPage.submeter();

        String mensagemErro = formularioPage.getMensagemDeErroVisivel();

        assertTrue(mensagemErro.contains("O nome é obrigatório."),
                "A mensagem de erro de nome obrigatório não foi encontrada.");
```

```
    }
}
```

- Dependências (Maven):
- pom.xml

```
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!--
https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
        <dependency>
            <groupId>io.github.bonigarcia</groupId>
            <artifactId>webdrivermanager</artifactId>
            <version>6.1.0</version>
        </dependency>
        <!--
https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>4.22.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
```

```
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

3. Evidências de Execução (PRINTS)

Abaixo estão os logs separados por tipo de teste.

3.1 Testes Unitários (SUCESSO)

Comando: `mvn -Dtest=EventoServiceTest test`

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.290 s -- in com.cli
ente.projeto.crudpb.service.EventoServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.505 s
[INFO] Finished at: 2025-12-08T17:53:17-03:00
[INFO] ------------------------------------------------------------------------
willump@lucas-popos:/mnt/Linux_Outros/Desenvolvimento/INFNET/PB/TP3/TP3_CODIGO
```

3.2 Testes de Integração (BUG ENCONTRADO)

Comando: mvn -Dtest=EventoControllerIntegrationTest test

```
./mvnw -Dtest=EventoControllerIntegrationTest test
[INFO]
[ERROR] Failures:
[ERROR]   EventoControllerIntegrationTest.deveRetornarErroNoFormulario_ParaEntradasInvalidas
:43 Status expected:<200> but was:<302>
[INFO]
[ERROR] Tests run: 4, Failures: 1, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.583 s
[INFO] Finished at: 2025-12-08T17:58:23-03:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:3.5.4:test (de
fault-test) on project com-cliente-projeto: There are test failures.
```

Evidência de que o teste detectou um comportamento inesperado (redirecionamento 302 ao invés de exibir erro na mesma página).

3.3 Testes E2E (SUCESSO - Firefox)

Comando: mvn -Dtest=CadastroEventoE2ETest test Teste de ponta-a-ponta executado com sucesso utilizando FirefoxDriver (Headless).

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 9.992 s -- in com.cli
ente.projeto.crudpb.e2e.pageobjects.CadastroEventoE2ETest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  12.834 s
[INFO] Finished at: 2025-12-08T17:59:47-03:00
[INFO] ------------------------------------------------------------------------
```

4. Conclusão TP3

O projeto atende aos requisitos de compilação, estrutura e cobertura de testes. A suíte de testes mostrou-se eficaz ao validar a lógica de negócios (Unitários), o fluxo de ponta-a-ponta (E2E) e identificar divergências de comportamento na camada web (Integração).