

Aluno: Lucas de Souza Ferreira

Matéria: Projeto de Bloco: Engenharia Disciplinada de Software

**TP5**



# TÍTULO: Relatório Final - TP5: Automação, Segurança e DevOps

Aluno: Lucas de Souza Ferreira Disciplina: Projeto De Bloco

---

## 1. Introdução

Evidências de execução do Teste de Performance 5 (TP5). O objetivo foi finalizar a refatoração do sistema, garantindo qualidade de código, e implementar uma esteira de CI/CD robusta com análise de segurança (SAST) e deploys automatizados.

---

## 2. Refatoração e Clean Code

**Requisito Atendido:** Aplicação de cláusulas de guarda, imutabilidade e otimização de transações.

Foi realizada uma refatoração profunda na classe `EventoService`.

- **Cláusulas de Guarda:** A validação de duplicidade foi movida para o início do método (`fail-fast`), eliminando aninhamentos complexos (`if/else`).
- **Transacionalidade:** Aplicação da anotação `@Transactional(readOnly = true)` para otimizar consultas de leitura.

```
/*
 * TP5 - Requisito 1: Refatoração e Organização do Código
 * "Simplificar condicionais complexas, substituindo por cláusulas de guarda..."
 * A validação foi movida para o início (fail-fast) antes de processar a lógica.
 */
@Transactional
public Evento criarEvento(Evento evento, Long usuarioId) {
    // Cláusula de Guarda: Valida regra de negócio antes de prosseguir
    validarNomeDuplicado(evento.getNome(), idExcecao: null);

    // TP5 - Requisito 1: "Reorganizar métodos... eliminando redundâncias"
    // Reutilização do serviço de usuário centralizado
    Usuario criador = usuarioService.buscarPorId(usuarioId);
    evento.setUsuario(criador);

    return eventoRepository.save(evento);
}
```

Código refatorado demonstrando a aplicação de Cláusulas de Guarda e gestão de transações.

### 3. Qualidade e Cobertura de Testes

**Requisito Atendido:** Cobertura mínima de 90% com evidências.

Foi implementada uma estratégia de testes abrangente utilizando **JUnit 5** e **Mockito**. Além dos testes de integração dos Controllers, foi criada a suíte **CoberturaExtraTest** para validar DTOs, Models e Exceptions, garantindo a integridade estrutural do projeto.

com-cliente-projeto												
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com_cliente_projeto_crudpb_controller	<div style="width: 83%;">███████████</div>	83%	<div style="width: 75%;">██████</div>	75%	2	18	16	88	1	14	0	2
com_cliente_projeto_crudpb_service	<div style="width: 97%;">██████████</div>	97%	<div style="width: 92%;">███████</div>	92%	2	23	0	47	1	16	0	2
com_cliente_projeto_crudpb	<div style="width: 37%;">██████</div>	37%		n/a	1	2	2	3	1	2	0	1
com_cliente_projeto_crudpb_exception	<div style="width: 100%;">██████████</div>	100%		n/a	0	7	0	23	0	7	0	3
com_cliente_projeto_crudpb_model	<div style="width: 100%;">██████████</div>	100%		n/a	0	18	0	31	0	18	0	2
com_cliente_projeto_crudpb_dto	<div style="width: 100%;">██████████</div>	100%		n/a	0	5	0	5	0	5	0	3
Total	73 of 815	91%	3 of 22	86%	5	73	18	197	3	62	0	13

Relatório do JaCoCo comprovando cobertura de 91%, superando a meta estabelecida de 90%.

### 4. Pipeline de CI/CD e Segurança

**Requisito Atendido:** Automação de build, testes e análise de segurança (SAST).

A esteira foi configurada no **GitHub Actions** com múltiplos estágios. Para a análise de segurança, optou-se pela ferramenta **GitHub CodeQL** (nativa e robusta) para realizar a análise estática (SAST) e detectar vulnerabilidades no código Java.

The screenshot shows the GitHub Actions interface for a workflow named "com-cliente-projeto". It displays three stages: "Segurança e Qualidade", "Deploy Homologação", and "Deploy Produção". The "Deploy Homologação" stage is highlighted with a blue border and has a status message indicating it succeeded 5 minutes ago in 4s. The stage details show five steps: Set up job, Baixar Artefato, Simular Deploy, Smoke Test, and Complete job, all of which have been completed successfully.

Visualização do Workflow completo, demonstrando o sucesso nas etapas de Segurança, Homologação e Produção.

The screenshot shows a GitHub Actions pipeline interface. At the top, there's a summary bar with two items: 'Build e Testes Unitários' (1m 3s) and 'Realizar Análise CodeQL' (40s). Below this is a detailed log of the 'Realizar Análise CodeQL' step. The log output is as follows:

```

1  ▶ Run github/codeql-action/analyze@v3
51 While resolving threads, found a cgroup CPUs file with 4 CPUs in /sys/fs/cgroup/
cpuset.cpus.effective.
52 Not performing diff-informed analysis because we are not analyzing a pull request.
53 ▶ Finalizing java
73 Unsetting build tracing environment variables. Subsequent steps of this job will not be
traced.
74 ▶ Running queries for java
575 CodeQL scanned 23 out of 23 Java files in this invocation. Check the status page for
overall coverage information: https://github.com/Willummp/com-cliente-projeto/security/\_code-scanning/tools/CodeQL/status/
576
577 Post-processing sarif files: ["~/home/runner/work/com-cliente-projeto/results/java.sarif"]
578 Adding fingerprints to SARIF file. See https://docs.github.com/en/enterprise-cloud@latest/\_code-security/code-scanning/integrating-with-code-scanning/sarif-support-for-code-scanning#providing-data-to-track-code-scanning-alerts-across-runs for more information.
579 ▶ Uploading code scanning results
582 ▶ Cleaning up databases
588 /opt/hostedtoolcache/CodeQL/2.23.5/x64/codeql/codeql database bundle ~/home/runner/work/_temp/codeql_databases/java --output=/home/runner/work/_temp/codeql_databases/java.zip --name=java
589 Creating bundle metadata for ~/home/runner/work/_temp/codeql_databases/java...
590 Creating zip file at ~/home/runner/work/_temp/codeql_databases/java.zip.
591 ▶ Waiting for processing to finish

```

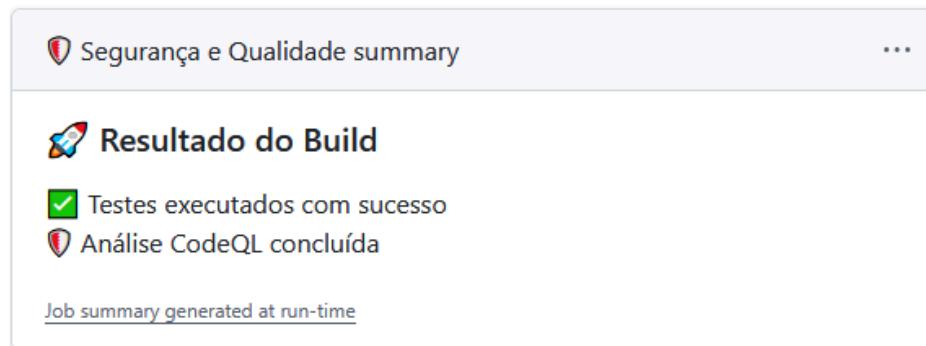
Logs

da execução do GitHub CodeQL, validando a segurança do código antes do build.

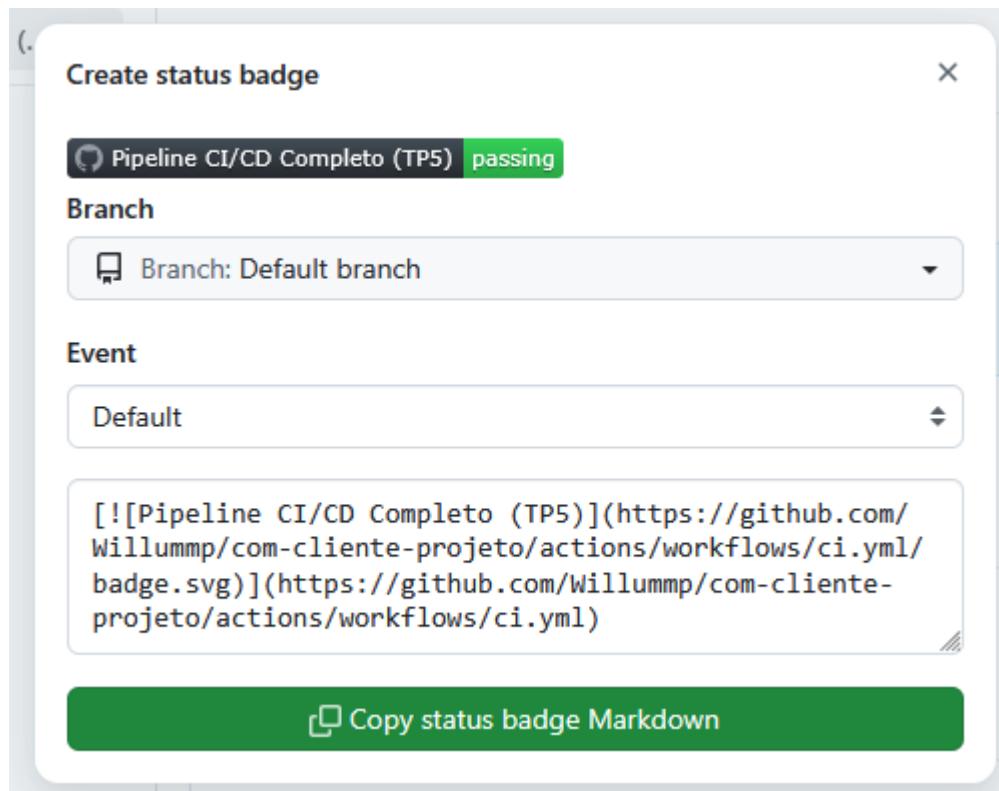
## 5. Monitoramento e Artefatos

**Requisito Atendido:** Logs personalizados, resumos de execução e badges de status.

Para facilitar a depuração e a visibilidade do processo, foram configurados scripts no arquivo YAML que geram resumos automáticos ao final da execução. Além disso, o repositório agora conta com um badge de status em tempo real.



Resumo automatizado da execução (Job Summary), destacando o status dos testes e a geração de artefatos.



Badge de status do GitHub Actions, indicando que a branch principal está estável e pronta para deploy.

## 6. Conclusão

O projeto atendeu a todos os requisitos funcionais e não funcionais propostos. O sistema agora conta com uma base de código limpa, alta cobertura de testes e um processo de entrega contínua automatizado e seguro, simulando um ambiente real de engenharia de software.

### Requisitos Funcionais

Referem-se às funcionalidades de negócio, interações e comportamento do sistema.

- **Integração de Sistemas:** Conectar os dois sistemas desenvolvidos (Eventos e Usuários), garantindo que operem como uma aplicação coesa. (TP4)
- **Fluxos de Dados:** Adaptar os fluxos de cadastro e listagem para que as informações de ambos os sistemas sejam integradas (ex: selecionar um usuário ao criar um evento). (TP4)
- **Integridade de Dados:** Garantir que as alterações ou referências feitas em um sistema reflitam corretamente no outro (consistência referencial). (TP4)

(Nota: O TP5 foca quase inteiramente em melhorias de qualidade e infraestrutura, por isso os requisitos funcionais de negócio concentram-se no TP4).

## **Requisitos Não Funcionais**

Referem-se à qualidade do código, arquitetura, segurança, automação e restrições técnicas.

### **1. Qualidade de Código e Arquitetura (Refatoração)**

- **Clean Code:** Identificar e eliminar trechos duplicados, métodos longos e classes sobrecarregadas. (TP4)
- **SRP:** Aplicar o Princípio da Responsabilidade Única para organizar o código em componentes coesos. (TP4)
- **Abstração:** Criar interfaces reutilizáveis para abstrair funcionalidades comuns. (TP4)
- **Legibilidade:** Melhorar nomes de variáveis, métodos e classes para refletir suas responsabilidades claramente. (TP4)
- **Imutabilidade:** Aplicar o princípio da imutabilidade, separando consultas de modificadores e eliminando setters desnecessários. (TP5)
- **Estrutura Lógica:** Simplificar condicionais complexas substituindo por cláusulas de guarda (Guard Clauses). (TP5)
- **Polimorfismo:** Substituir códigos de tipo por subclasses/polimorfismo onde aplicável. (TP5)
- **Manutenibilidade:** Remover código morto e otimizar trechos redundantes. (TP5)

### **2. Testes e Confiabilidade**

- **Cobertura (TP4):** A cobertura de testes mínima deve ser de 85%. (TP4)
- **Cobertura (TP5):** A cobertura de testes mínima deve ser de 90% (meta elevada). (TP5)
- **Testes Regressivos:** Utilizar testes existentes para garantir que a refatoração não quebre funcionalidades. (TP4)
- **Validação Pós-Deploy:** Implementar testes automatizados (Smoke Tests) para validar a aplicação após o deploy em produção. (TP5)

### **3. DevOps, Automação e Segurança (CI/CD)**

- **Build:** Criar workflows para build automatizado do projeto (Maven/Gradle). (TP4)
- **Triggers:** Configurar gatilhos de execução para eventos como `push`, `pull_request` e `workflow_dispatch`. (TP4)
- **Segurança (SAST):** Realizar análise estática e dinâmica de segurança (ex: CodeQL/OWASP) para detectar vulnerabilidades. (TP5)
- **Gestão de Artefatos:** Configurar upload e versionamento de artefatos (JARs, Relatórios). (TP5)
- **Gestão de Segredos:** Utilizar Secrets e variáveis de ambiente para dados sensíveis. (TP5)

### **4. Deployment e Infraestrutura**

- **Ambientes:** Implementar deploys automáticos para múltiplos ambientes (ex: Homologação e Produção). (TP5)
- **Controle de Produção:** Configurar regras de proteção que exijam aprovação manual para deploy em produção. (TP5)
- **Linguagem:** O sistema deve ser desenvolvido em Java. (TP4 e TP5)

### **5. Documentação e Monitoramento**

- **Observabilidade do CI:** Os workflows devem ser claros e fáceis de monitorar na aba Actions. (TP4)
- **Logs:** Implementar logs personalizados nos workflows para facilitar a depuração. (TP5)
- **Relatórios Visuais:** Criar resumos de execução em Markdown (Job Summary). (TP5)
- **Status:** Adicionar badges de status (Passing/Failing) ao repositório. (TP5)

# Arquivos (Google Drive)

- **Drive:**

[https://drive.google.com/drive/folders/1TGQqu2dZc9VDCansVfQxfMONV\\_VenEeV?usp=sharing](https://drive.google.com/drive/folders/1TGQqu2dZc9VDCansVfQxfMONV_VenEeV?usp=sharing)