CESAR School

**Especialização em Liderança Técnica**

# JORNADA FINAL DEVSECOPS NA PRÁTICA

**Autores:**
Douglas Azevedo
Gustavo Nicolau Jacintho
José Augusto Alves
Pedro Williams

Outubro/2025

# INTRODUÇÃO

A finalidade principal desta documentação é demonstrar a visão estratégica e organizacional da adoção do DevSecOps na SecureTasks. Este documento fornece a justificativa de negócio para a transformação cultural e técnica, estabelecendo a Política Mínima de Segurança da Informação que guiará a empresa.

A SecureTasks é uma startup promissora no setor de soluções digitais, mas enfrenta um desafio crítico: sua aplicação web pública, utilizada por milhares de usuários, contém falhas graves de segurança. A organização, embora possua uma equipe de desenvolvimento experiente, opera historicamente sem práticas formais de segurança. Este vácuo é agravado por uma alta gestão que não acompanha o dia a dia técnico, criando um ponto cego perigoso.

Este cenário representa um risco de negócio iminente, afetando não apenas a confiança do cliente e a reputação da marca, mas também expondo a empresa a potenciais perdas financeiras e sanções legais. A inação deixou de ser uma opção.

## Objetivo do Projeto e da Transformação DevSecOps

Diante deste risco existencial, a SecureTasks inicia uma jornada de transformação DevSecOps. O objetivo deste projeto é simular os passos práticos para construir a fundação dessa transformação.

Não se trata apenas de "comprar ferramentas", mas de redesenhar o ciclo de vida de desenvolvimento de software (SDLC) para que a segurança seja integrada, automatizada e transparente.

Os objetivos centrais desta iniciativa são:

1. **Automatizar a Detecção de Vulnerabilidades (*Security Gates*)** Garantir que falhas sejam identificadas automaticamente no primeiro *commit*, e não semanas depois. Isto será alcançado pela integração de um arsenal de ferramentas de análise diretamente no pipeline de CI/CD (GitHub Actions), incluindo:
    - **Análise de Composição de Software (SCA):** Para identificar vulnerabilidades em dependências de terceiros.
    - **Análise Estática de Segurança (SAST):** Para encontrar falhas no código-fonte antes da execução.
    - **Varredura de Segredos (*Secret Scanning*):** Para impedir o vazamento de chaves de API, senhas e licenças.
    - **Varredura de Imagens de Contêiner:** Para analisar a segurança das imagens Docker antes do deploy.
2. **Habilitar o Mapeamento e a Gestão de Riscos** Mudar da postura de "não sabemos o que não sabemos" para uma gestão de risco informada. Os relatórios gerados pelas ferramentas de scan (centralizados em plataformas como *SonarQube* e *ArcherySec*) fornecerão visibilidade clara das vulnerabilidades, permitindo que a gestão técnica priorize correções com base no impacto real.

3. **Fomentar a Cultura de Segurança (*Shift Left*)** Iniciar a mudança cultural onde a segurança é uma responsabilidade compartilhada. A automação e o feedback rápido, desde a estação do desenvolvedor (com *hooks de pre-commit*) até os relatórios nos Pull Requests, servem para educar a equipe e integrar as boas práticas ao fluxo de trabalho diário, muito antes do código chegar em produção.

# JUSTIFICATIVA ESTRATÉGICA

A adoção do DevSecOps é considerada uma estratégia de negócio fundamental para a SecureTasks, e não apenas uma iniciativa técnica. A existência de falhas críticas, que ameaçam a reputação da empresa e a confiança de seus milhares de usuários, justifica a urgência em integrar a segurança de forma fundamental ao ciclo de desenvolvimento.

O investimento nesta transformação se paga ao mudar a segurança de um "centro de custo" reativo para um "habilitador de valor" proativo. Os benefícios diretos desta integração, implementados neste projeto, são:

1. **Mitigação Proativa e "Shift Left" de Riscos** O DevSecOps permite que a segurança seja tratada de forma contínua, automatizada e, o mais importante, antecipada. Em vez de auditar a produção, garantimos que as vulnerabilidades sejam identificadas imediatamente.
    - **Na prática:** Isso começa na estação do desenvolvedor, com **hooks de pre-commit** (como o *Gitleaks*) que bloqueiam segredos antes mesmo de serem versionados. O processo continua nos Pull Requests, onde pipelines de CI/CD em **GitHub Actions** executam uma bateria de testes, incluindo **SAST** (*SonarQube*), **SCA** (*Dependency Track*), **Secret Scanning** (*Trivy*) e **Varredura de Imagens de Contêiner**. A identificação precoce reduz drasticamente o custo e o impacto da correção.
2. **Mapeamento e Gerenciamento de Riscos Baseado em Dados** O objetivo é substituir a incerteza por uma gestão de risco informada. A integração de ferramentas não serve apenas para "encontrar falhas", mas para centralizar, classificar e priorizar.
    - **Na prática:** As evidências de falhas são agregadas em dashboards centralizados (como **SonarQube** para código, **Dependency Track** para componentes e **ArcherySec** para scans de contêiner). Isso permite que a gestão de segurança visualize o *real* apetite a risco, utilize métricas (como CVSS e EPSS do *Dependency Track*) para priorizar correções e acompanhe a evolução da postura de segurança do produto ao longo do tempo, tudo integrado à aba "Security" do GitHub.
3. **Apoio à Agilidade Operacional e Entrega Contínua** Historicamente, a segurança em "cascata" (waterfall) é um gargalo que atrasa a entrega. O DevSecOps remove esse gargalo ao integrar a segurança ao mesmo fluxo ágil.
4. **Desenvolvimento de Maturidade Organizacional e Colaboração** A transformação DevSecOps é, acima de tudo, cultural. Ela quebra silos e estimula a colaboração entre

perfis técnicos e de gestão, criando um Ciclo de Vida de Desenvolvimento Seguro (SDLC) maduro.

- **Na prática:** O **Pull Request (PR)** torna-se o ponto central de colaboração. Nele, desenvolvedores, *scans* automatizados e revisores de segurança interagem. A pipeline de CI/CD, definida como código, torna-se a documentação viva do nosso processo de SDLC, garantindo consistência e transparência. Isso cria um ciclo de feedback rápido que educa a equipe e eleva o padrão de qualidade do software como um todo

Espera-se que, com a implementação prática dessas ferramentas e processos automatizados, a SecureTasks alcance maior confiabilidade, conformidade regulatória e uma redução drástica no número de incidentes de segurança. Tais resultados são essenciais para proteger a receita e manter a competitividade da empresa no setor de soluções digitais.

# DIRETRIZES DA POLÍTICA MÍNIMA DE SEGURANÇA

Esta política aplica-se a todos os colaboradores envolvidos no ciclo de desenvolvimento e infraestrutura (Dev, Sec e Ops). Ela é mandatória para garantir a integração efetiva das verificações de segurança em todo o ciclo de vida, desde o primeiro *commit* de código até a implantação em produção.

O princípio fundamental é o **Shift Left**: a segurança é uma responsabilidade compartilhada e deve ser automatizada e antecipada.

## Princípios Fundamentais

1. **Segurança como Responsabilidade Compartilhada:** A segurança não é uma função isolada de um time de "auditores". Ela é um compromisso de toda a equipe técnica e deve ser apoiada pela gestão.
2. **Automação como Padrão (CI/CD):** É obrigatório o uso dos pipelines de CI/CD configurados no **GitHub Actions**. Todo o processo de detecção de falhas de segurança deve ser automatizado, e falhas críticas (conforme definido na metodologia de risco) devem atuar como *security gates*, bloqueando o *merge* de código inseguro.
3. **Implementação Mínima de Técnicas:** A SecureTasks adota, no mínimo, as seguintes técnicas de scan automatizado, conforme implementado neste projeto: SAST, SCA, *Secret Scanning* e *Container Scanning*.
4. **Geração e Centralização de Evidências:** Os *scans* de segurança devem gerar relatórios auditáveis. As evidências de falhas devem ser publicadas automaticamente nos *Pull Requests* (na aba "Checks" e "Security") e centralizadas nos *dashboards* das ferramentas de gestão (como **SonarQube**, **Dependency Track** e **ArcherySec**).

## Controles Mandatórios no Ciclo de Vida (SDLC)

O ciclo de desenvolvimento de software da SecureTasks deve integrar as seguintes verificações de segurança de forma contínua:

## 1. Na Estação de Trabalho do Desenvolvedor (Pre-Commit)

Antes que o código seja enviado ao repositório central, o desenvolvedor deve ter os *hooks* de *pre-commit* ativados.

- **Controle:** *Secret Scanning* local.
- **Ferramenta Implementada: Gitleaks**.
- **Objetivo:** Impedir que segredos (senhas, chaves de API, tokens) sejam "comitados" acidentalmente no histórico do Git.

## 2. Na Revisão de Código (Pull Request - CI)

Nenhum Pull Request (PR) pode ser mesclado na *branch principal* (`main`) sem a execução e aprovação dos seguintes *scans* automatizados via **GitHub Actions**:

- **Controle: Análise Estática (SAST)**
  - **Descrição:** Análise do código-fonte em busca de padrões de vulnerabilidade (ex: SQL Injection, XSS, falhas de lógica).
  - **Ferramenta Implementada: SonarQube**.
- **Controle: Análise de Composição (SCA)**
  - **Descrição:** Varredura das dependências do projeto (*package.json*) em busca de bibliotecas de terceiros com vulnerabilidades conhecidas (CVEs).
  - **Ferramenta Implementada: Dependency Track** e **Trivy**.
- **Controle: Varredura de Segredos (Secret Scanning)**
  - **Descrição:** Uma segunda camada de defesa (além do *pre-commit*) para detectar segredos expostos no código.
  - **Ferramenta Implementada: Gitleaks** e **Trivy**.

## 3. Na Geração de Artefatos (Build)

Durante o processo de build da imagem de aplicação, o pipeline deve executar a varredura do artefato gerado.

- **Controle: Varredura de Imagem de Contêiner**
  - **Descrição:** Análise da imagem Docker final em busca de vulnerabilidades no sistema operacional base (ex: *Ubuntu*, *Alpine*) e em pacotes de sistema, além de más configurações.
  - **Ferramenta Implementada: Trivy**.

# POLÍTICA DE SEGURANÇA – GOVERNANÇA E RESPONSABILIDADES

A responsabilidade pela segurança é distribuída entre os perfis estratégicos e técnicos da organização, conforme a seguinte especificação:

| Área/Perfil Responsável | Responsabilidade Chave |
|---|---|
| Gestão de Segurança da Informação (GSI) | Responsável pela definição da estratégia, produção de documentação formal, e validação da metodologia de risco |
| Responsaúeis pela Criação de Pipelines/Infraestrutura | Responsáveis pela automação de processo de detecção de falhas, implementação das pipelines CI/C com scans, e gestão do código de infraestrutura (Iac) |
| Times de Desenvolvimento e Operação (Dev/Ops) | Responsáveis por aplicar as diretrizes e boas práticas mínimas estabelecidas e pela correção das falhas identificadas nos relatórios de scans |
| Alta Gestão (C-level) | Responsáveis por fornecer o apoio estratégico e garantir que os recursos sejam alocados para a transformação DevSecOps |

O monitoramento e a resposta a vulnerabilidades devem ser contínuos. As vulnerabilidades identificadas pelas ferramentas de scan devem ser validadas pela GSI em conjunto com os times técnicos.

A priorização da correção deve ser baseada na metodologia de avaliação de risco adotada pela SecureTasks. O ambiente DevSecOps existe monitoramento contínuo da execução das pipelines CI/CD e dos scans.

# METODOLOGIA DE AVALIAÇÃO E MITIGAÇÃO DE RISCOS

A SecurTasks adota uma metodologia de Análise de Risco Qualitativa, focada na combinação da probabilidade de exploração de uma falha e seu impacto potencial no Negócio. Esta metodologia fornece clareza para a priorização de correções pelas equipes de desenvolvimento e operações.

Todas as vulnerabilidades detectadas  pelas pipelines CI/CD devem ser classificadas em relação a dois fatores:

1. **Probabilidade:** Classifica a facilidade com que a vulnerabilidade pode ser explorada.

2. **Impacto:** Classifica o dano potencial ao negócio, sendo eles: financeiro, legal, reputacional e operacional.

O nível de risco é determinado pela combinação dos fatores de probabilidade e impacto, conforme a tabela de prioprização abaixo:

| Probabilidade (P) | Impacto Alto (A) | Impacto Médio (M) | Impacto Baixo (B) |
|---|---|---|---|
| Alta (A) | Risco Crítico (C) | Risco Alto (A) | Risco Médio (M) |
| Médio (M) | Risco Alto (A) | Risco Médio (M) | Risco Baixo (B) |
| Baixa (M) | Risco Médio (M) | Risco Baixo(B) | Risco Tolerável (T) |

Vulnerabilidades classificadas como Risco Crítico (Combinação Alta/Alta) exigem tratamento imediato e devem ser corrigidas antes de qualquer nova funcionalidade.

## Tratamento e Resposta

Para cada nível de risco, a equipe responsável deve definir um plano de tratamento, que pode incluir remediação, mitigação ou aceitação formal. A eficácia do tratamento deve ser verificada pela reexecução dos scans automatizados.

# CONCLUSÃO: O PRIMEIRO PASSO NA JORNADA

A SecureTasks iniciou este projeto em um estado de alto risco: uma aplicação web pública com falhas críticas, uma equipe de desenvolvimento experiente, mas operando sem processos formais de segurança, e uma alta gestão desalinhada da realidade técnica. O desafio era claro: transformar a segurança de um gargalo reativo para um habilitador de agilidade.

Este documento de gestão, em conjunto com o repositório técnico associado (https://github.com/doug-cpp-devsecops/proj-final), representa a fundação prática e estratégica dessa transformação.

## Conseguimos provar que é possível:

1. **Automatizar a Detecção:** A implementação dos *workflows* de CI/CD no GitHub Actions, utilizando um arsenal de ferramentas (*Gitleaks, SonarQube, Dependency Track, Trivy, Checkov*), é a prova de que a detecção de vulnerabilidades pode (e deve) ser automática, removendo o fardo do esforço manual e fornecendo feedback instantâneo.

2. **Conectar Gestão e Técnica:** A "Política Mínima de Segurança" e a "Metodologia de Avaliação de Risco" aqui descritas não são teóricas. Elas são o manual de governança que direciona o uso das ferramentas implementadas. Os relatórios gerados pelas ferramentas são a entrada direta para a nossa matriz de risco, permitindo que a gestão tome decisões informadas.
3. **Manter a Agilidade:** Através da Infraestrutura como Código (*Kubernetes* e *Docker*), demonstramos que a infraestrutura pode ser criada e validada com a mesma velocidade do código da aplicação, garantindo um processo de *deploy* fluido e seguro.

O objetivo deste projeto da disciplina de DevSecOps foi *construir o motor* de detecção e *definir o mapa* de governança. A jornada da SecureTasks, no entanto, apenas começa.

Os relatórios de *scan* gerados (Entregável 4.a) agora formam o *backlog* de segurança inicial da empresa. Os próximos passos são claros: aplicar a metodologia de risco para priorizar essas falhas, iniciar o ciclo de remediação e, o mais importante, solidificar a cultura de responsabilidade compartilhada que esta política estabelece.

Ao adotar esta abordagem DevSecOps, a SecureTasks deixa de ser reativa e passa a ser proativa. A empresa agora possui uma fundação documentada, automatizada e estratégica para entregar software de forma ágil *e* segura, protegendo seus usuários, sua reputação e sua competitividade no mercado de soluções digitais.