TASK FLOW

Plano de Gerenciamento de Configuração

Renato Matos, Felipe Picinin, Lucas Garcia, Renato Cazzoleti e Pedro H. Braga

INTRODUÇÃO

O **Plano de Gerenciamento de Configuração** estabelece as diretrizes fundamentais para o controle sistemático de todos os artefatos do projeto TaskFlow. Ele define práticas rigorosas de identificação, controle de mudanças, auditorias e manutenção da integridade.

O **TaskFlow** é uma API RESTful desenvolvida em Node.js e TypeScript, criada para aplicar na prática os conceitos de Gerência de Configuração. O sistema oferece funcionalidades completas de gestão de tarefas com alta confiabilidade.

OBJETIVOS

Diretrizes e Padronização

Estabelecer responsabilidades claras, ferramentas adequadas e métodos padronizados para garantir que toda a equipe siga as mesmas práticas.

Consistência do Ciclo de Vida

Assegurar a consistência dos artefatos desde a concepção até a entrega, mantendo conformidade com os padrões da instituição.

Rastreabilidade e Qualidade

Permitir a identificação da origem das mudanças e garantir que apenas código testado e aprovado seja integrado ao produto final.

ESCOPO DO PLANO

Código e Arquitetura

- Código-fonte da API RESTful
- Camada de Models e Repositories
- Services e Regras de Negócio
- Controllers e Rotas
- Controle rigoroso de versionamento

Automação e Qualidade

- Scripts de configuração
- Testes Unitários (Jest)
- Testes de Integração e Aceitação
- Pipeline de CI/CD (GitHub Actions)
- Automação de Build e Deploy

Documentação e Entrega

- Especificação OpenAPI/Swagger
- Dockerfile e Containerização
- Orquestração com Docker Compose
- Artefatos de Release
- Documentação Técnica

PROBLEMAS E DESAFIOS

COORDENAÇÃO DE EQUIPE

O desenvolvimento colaborativo com cinco membros exige sincronização precisa. Sem processos definidos, alterações simultâneas geram conflitos de código e inconsistências frequentes.

GARANTIA DE QUALIDADE

É crítico assegurar que todas as mudanças sejam testadas e revisadas antes da integração. A falta de validação pode introduzir bugs e comprometer a estabilidade do sistema em produção.

RASTREABILIDADE E CONTROLE

Manter o histórico completo de alterações e um controle rigoroso de versões é um desafio operacional. A ausência de estrutura resulta em perda de histórico e dificuldade em auditorias.

REVISÃO DE LITERATURA

Fundamentação Teórica (IBM RUP)

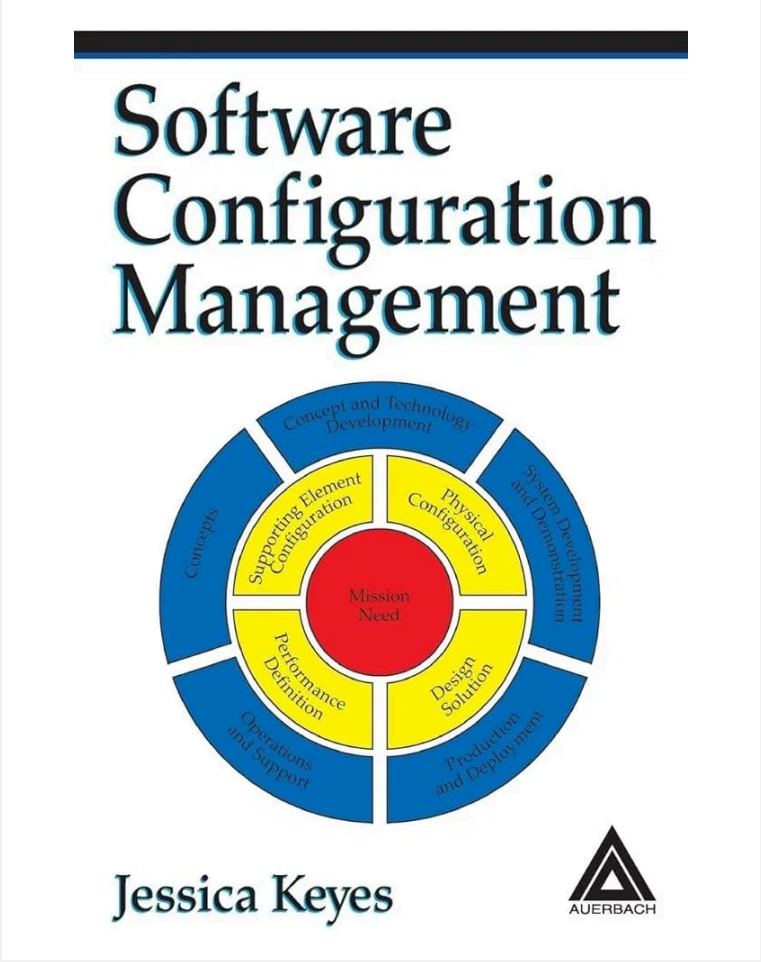
O plano baseia-se no **Template do Plano de Gerenciamento de Configuração RUP 7.0** da IBM, um padrão consolidado na indústria que fornece uma estrutura robusta para gestão de configuração.

Padrões de Comunidade

Adoção de **Conventional Commits** para padronização de mensagens e **Versionamento Semântico** (MAJOR.MINOR.PATCH) para facilitar a automação e o rastreamento histórico.

Documentação Técnica Moderna

Utilização das documentações oficiais do **GitHub Actions**, **Docker** e **Jest** para implementar pipelines de CI/CD e containerização com as melhores práticas atuais.



ORGANIZAÇÃO DA EQUIPE

GERENTE DE CONFIGURAÇÃO

Lucas Garcia

Responsável por manter o plano, definir políticas de branching e versionamento, e orientar a equipe nas decisões estratégicas.

DESENVOLVEDOR DA API

Renato

Responsável pelo desenvolvimento do código-fonte da API e garantia de conformidade com os padrões definidos.

RESPONSÁVEL POR TESTES

Renato Cazzoleti

Gerenciamento e execução de testes unitários, de integração e aceitação para garantir a qualidade do software.

RESPONSÁVEL PELO PIPELINE

Pedro Braga

Desenvolvimento, manutenção e otimização do pipeline de CI/CD para automação de build e deploy.

RESPONSÁVEL PELA CONTAINERIZAÇÃO

Felipe Picinin

Criação e manutenção dos artefatos Docker e gestão dos ambientes containerizados para reprodutibilidade.

FERRAMENTAS E AMBIENTES

STACK TECNOLÓGICO

Core: Node.js (v18+), TypeScript, Express.js

Qualidade: Jest (Testes), ESLint

Infra: Docker, Git, GitHub Actions

AMBIENTES CONTROLADOS

Desenvolvimento: Local com Hot-reload

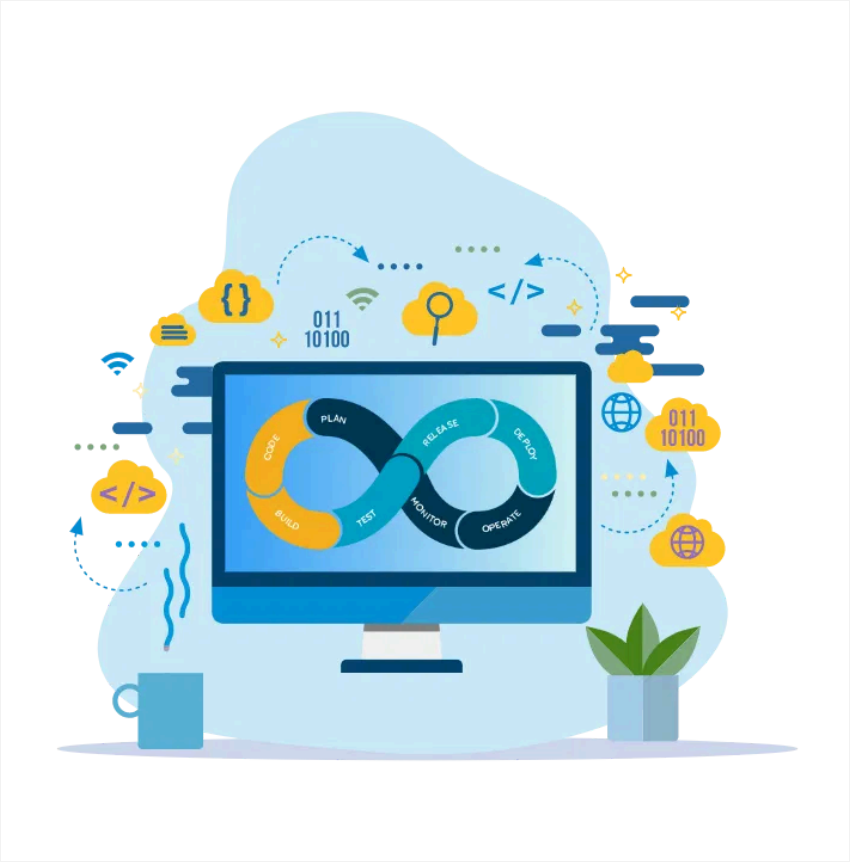
Integração: CI via GitHub Actions

Produção: Containerizado em Docker

ESTRATÉGIA DE BRANCHING

main (produção), develop (integração)

feature/*, bugfix/*, hotfix/*



PROCESSO DE CONTROLE DE MUDANÇAS

FLUXO DE MUDANÇAS

1. Criação de branch (develop)
2. Desenvolvimento e commits
3. Execução de testes locais
4. Abertura de Pull Request
5. Revisão de código
6. Pipeline automatizado
7. Merge após aprovação

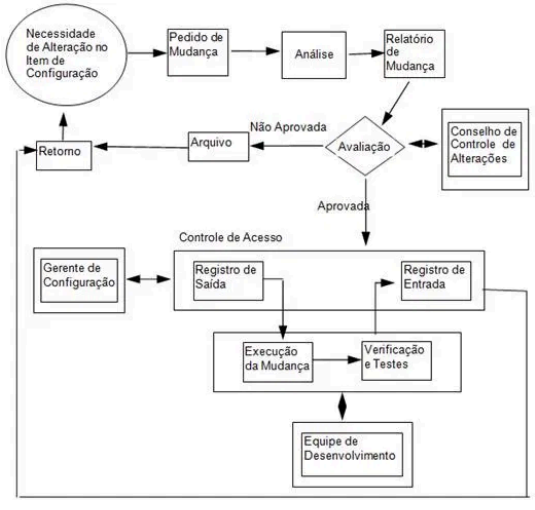
NÍVEIS DE APROVAÇÃO (CCB)

Correção de Bugs
1 Revisor

Nova Funcionalidade
2 Revisores

Alteração Arquitetural
Toda a Equipe

Pipeline
Responsável + 1



BASELINES, RELEASES E AUDITORIAS

DISCUSSÃO E BENEFÍCIOS

CONCLUSÃO

Fundamentos Sólidos

O Plano de Gerenciamento de Configuração garantiu a **qualidade**, **rastreabilidade** e **consistência** de todos os artefatos. A estruturação clara de papéis e processos foi crucial para a colaboração efetiva da equipe.

Aplicação Prática

A adoção de ferramentas modernas e práticas da indústria (Git, Docker, CI/CD) preparou a equipe para desafios reais. O projeto demonstrou a importância vital da gerência de configuração no sucesso do software.

Resultados Alcançados

70%+

Cobertura de Testes

100%

Pipeline Automatizado

v1.0

Release Containerizada