TASK FLOW
Plano de Gerenciamento de Configuração

Versão 1.0

Histórico da Revisão

Data	Versão	Descrição	Autor
01/12/2025	1.0	Primeira versão	Felipe Picinin

Índice

Plano de Gerenciamento de Configuração

1. Introdução

O Plano de Gerenciamento de Configuração descreve todas as atividades relacionadas ao controle e à gestão da configuração do projeto TaskFlow, abrangendo identificação, controle de mudanças, auditorias e manutenção da integridade dos itens de configuração ao longo de seu ciclo de vida.

1.1 Objetivo

O objetivo deste Plano de Gerenciamento de Configuração é estabelecer diretrizes, responsabilidades, ferramentas e métodos a serem seguidos pelos membros da equipe de desenvolvimento do TaskFlow, garantindo consistência, rastreabilidade e qualidade dos artefatos produzidos.

1.2 Escopo

Este plano aplica-se a todos os integrantes da equipe responsável pelo desenvolvimento do TaskFlow, projeto realizado como atividade prática da disciplina de Gerência de Configuração e Evolução de Software da PUC Minas.

O escopo abrange o controle de itens como:
– código-fonte da API;
– scripts de configuração;
– testes automatizados;
– pipeline de integração contínua;
– documentação;
– artefatos de entrega e containerização.

1.3 Definições, Acrônimos e Abreviações

GCS – Gerência de Configuração de Software
CI – Continuous Integration
CD – Continuous Delivery
API – Application Programming Interface
REST – Representational State Transfer
Baseline – Conjunto de versões estáveis de itens de configuração
Branch – Ramificação no repositório
Commit – Registro de alteração
Pull Request – Solicitação de integração entre branches
Pipeline – Fluxo automatizado de build, testes e deploy
Docker – Plataforma de containerização
Jest – Ferramenta de teste para JavaScript/TypeScript

1.4 Referências

Template do Plano de Gerenciamento de Configuração RUP 7.0 da IBM;

Documento de Requisitos do Trabalho Prático – PUC Minas;
Documentação oficial do GitHub Actions;
Documentação oficial do Docker e do Jest.

1.5 Visão Geral

Este documento está dividido em:
– Seção 2: organização da equipe e infraestrutura;
– Seção 3: identificação, controle e auditoria da configuração;
– Seção 4: marcos;
– Seção 5: treinamento e recursos;
– Seção 6: controle de software externo.

2. Gerenciamento de Configuração de Software

2.1 Organização, Responsabilidades e Interfaces

Gerente de Configuração – Lucas Garcia: responsável por manter este plano, definir políticas de branching e versionamento e orientar a equipe.

Desenvolvedor da API – Renato: responsável pelo código-fonte e conformidade com padrões definidos.

Responsável por Testes – Renato Cazzoletti: desenvolvimento e manutenção dos testes unitários, de integração e aceitação.

Responsável pelo Pipeline – Pedro Braga: desenvolvimento e manutenção do pipeline CI/CD.

Responsável pela Containerização – Felipe Picinin: criação e manutenção dos artefatos de Docker e ambientes containerizados.

2.2 Ferramentas, Ambiente e Infra-estrutura

As ferramentas utilizadas incluem Git, GitHub, Node.js, TypeScript, Express, Jest, Visual Studio Code, Docker e GitHub Actions.

Os ambientes compreendem:
– Ambiente de Desenvolvimento local;
– Ambiente de Integração contínua via GitHub Actions;
– Ambiente Containerizado em Docker.

O repositório segue estrutura organizada em branches main, develop, feature/*, bugfix/* e hotfix/*.

3. O Programa de Gerenciamento de Configuração

3.1 Identificação da Configuração

3.1.1 Métodos de Identificação

Os arquivos são nomeados segundo convenções de nomenclatura definidas (PascalCase para classes, camelCase para funções e variáveis).

Os commits seguem o padrão "Conventional Commits".

As versões seguem o padrão semântico: vMAJOR.MINOR.PATCH.

3.1.2 Linhas de Base do Projeto

As baselines são definidas conforme etapas do projeto:

BL-1.0 – Desenvolvimento: código da API, models, repositories, services, controllers e rotas.
BL-2.0 – Testes: testes unitários, de integração e aceitação.
BL-3.0 – CI/CD: pipeline do GitHub Actions.
BL-4.0 – Release: Dockerfile, docker-compose e documentação.

3.2 Configuração e Controle de Alterações

3.2.1 Processamento e Aprovação de Controles de Mudanças

O fluxo de mudanças envolve:

1. criação de branch a partir de develop;
2. desenvolvimento e commits conforme padrão;
3. execução de testes locais;
4. abertura de Pull Request;
5. revisão de código;
6. execução do pipeline automatizado;
7. merge após aprovação.

3.2.2 CCB (Conselho de Controle de Mudanças)

O CCB é composto por toda a equipe, com níveis de aprovação conforme tipo de mudança:
Correção de bugs – 1 revisor;
Nova funcionalidade – 2 revisores;
Alteração arquitetural – todos;
Alterações no pipeline – responsável pelo pipeline + 1 revisor.

3.3 Contabilidade do Status de Configuração

3.3.1 Armazenamento de Mídia do Projeto e Processo de Release

O repositório GitHub funciona como armazenamento central e backup.

O processo de release compreende:
– merge de develop para main;
– criação de tag;
– execução do pipeline;
– geração de imagem Docker;
– publicação das notas de release.

Os conteúdos incluem código compilado, imagem Docker, documentação e changelog.

3.3.2 Relatórios e Auditorias

Os relatórios gerados pelo pipeline incluem:
– Cobertura de testes;
– Resultados de execução;
– Status de build;
– Análise de qualidade.

Auditorias verificam conformidade com padrões, cobertura de testes, funcionamento do pipeline e atualização da documentação.

4. Marcos

M1 – Semana 1: definição do projeto e criação do repositório.
M2 – Semana 2: desenvolvimento da API.
M3 – Semana 3: implementação dos testes.
M4 – Semana 4: configuração do pipeline CI/CD.
M5 – Semana 4: containerização.
M6 – Semana 5: documentação final e entrega.

5. Treinamento e Recursos

São necessários conhecimentos em Node.js, TypeScript, Express, Jest, GitHub Actions, Docker e Git.
Os recursos incluem acesso à documentação oficial das ferramentas utilizadas.

6. Controle de Software do Subfornecedor e do Fornecedor

O projeto utiliza bibliotecas de terceiros, todas devidamente licenciadas.
As dependências seguem políticas de atualização:
– atualizações de segurança são imediatas;
– atualizações major exigem aprovação;
– todas passam pelo pipeline de testes.