PLANO DE GERÊNCIA DE CONFIGURAÇÃO (CM)

1. Introdução

Este documento apresenta o Plano de Gerência de Configuração (Configuration Management – CM) para uma linha de produto composta por três variantes: low-end, medium e high-end. O objetivo é definir processos, políticas e estruturas que garantam controle, rastreabilidade, organização e qualidade na evolução dos artefatos de software, contemplando desenvolvimento contínuo, ciclos de lançamento e manutenção pós-deploy.

2. Objetivo

Estabelecer diretrizes formais de CM para o desenvolvimento e manutenção da linha de produtos, definindo como versões são controladas, como branches são organizadas, e como mudanças (change requests) e correções (bug fixes) são tratadas.

3. Escopo

Este plano cobre:
a) Organização de branches e fluxo de desenvolvimento.
b) Versionamento e empacotamento das variantes.
c) Tratamento de mudanças: *change requests* e correção de *bugs*.
d) Geração de *builds*, *releases* e *hotfixes*.
e) Rastreabilidade e controle formal dos artefatos.

4. Linha de Produto

A linha de produto é composta por três variantes geradas a partir de um único código-base:
a) Low-end: versão simplificada com conjunto mínimo de funcionalidades.
b) Medium: versão intermediária com funcionalidades adicionais.
c) High-end: versão completa com todos os módulos, recursos e otimizações.

Diferenciações de funcionalidades entre variantes são controladas por *feature toggles*, módulos configuráveis ou perfis de *build*.

5. Estratégia de *Branching*

A estratégia de branching adotada é baseada no modelo GitFlow, com adaptações específicas para a linha de produto.

5.1 Branches Principais

Existem duas *branches* principais que sustentam o fluxo de trabalho.

5.1.1 Desenvolvimento (develop)

A ***branch develop*** é responsável pela integração contínua. Todas as mudanças (*change requests* ou *bug fixes* não críticos) são inicialmente incorporadas nesta branch.

Ela representa o estado mais atual e instável do software, agregando funcionalidades preparadas para o próximo ciclo de *release*.

5.1.2 Lançamento/Produção (main)

A ***branch main*** representa o código estável em produção. Ela recebe atualizações apenas por meio de *merges* provenientes de:

- Branches de release, ao final de um ciclo de desenvolvimento.
- Branches de hotfix, quando há correções críticas que precisam ser entregues com urgência.

Nota sobre Branches de Preparação (release)

Quando o software atinge maturidade suficiente na develop para um novo lançamento, cria-se uma branch release (ex.: release/v1.2.0). Essa branch é utilizada para:

a) Testes finais
b) Estabilização
c) Correção de defeitos de última hora

Após finalizada, ela é:

1. Mesclada na main, onde recebe uma tag de versão.
2. Mesclada de volta na develop, garantindo alinhamento.

5.1.3 Branches Auxiliares

Branches auxiliares seguem o padrão:

a) feature/ para novas funcionalidades ou CRs.
b) hotfix/ para correções críticas que afetam a produção.
c) bugfix/ para defeitos não críticos encontrados no desenvolvimento.

Cada branch deve estar vinculada a um item de controle: CR, incidente, tarefa ou issue.

6. Política de Versionamento

A linha de produto adota versionamento semântico (Semantic Versioning), com três níveis:

- **MAJOR**: alterações incompatíveis que modificam significativamente o sistema;
- **MINOR**: inclusão de funcionalidades ou CRs entre releases;
- **PATCH**: correções de defeitos (geralmente provenientes de hotfixes);

6.1 Versionamento Unificado da Linha de Produto

Toda a linha de produto compartilha um único número de versão (ex.: v1.2.0), aplicado ao código-base. A diferenciação entre low-end, medium e high-end ocorre somente no momento do build.

6.2 Geração de Artefatos das Variantes

Durante o processo de build, aplicam-se:

a) Feature toggles
b) Perfis de build
c) Módulos específicos por variante

Isso garante que:

- Todas as variantes compartilhem o mesmo nível de correções PATCH.
- Novas funcionalidades MINOR possam ser ativadas somente nas variantes desejadas, sem fragmentar o código-base.
- A evolução permaneça síncrona entre produtos.

7. Processo para Change Requests (CR)

Change Requests afetam apenas o desenvolvimento. O ciclo é:

1. Registro e aprovação formal do CR.
2. Criação de branch feature/CR-xxx a partir de develop.
3. Implementação, testes e revisão.
4. Merge na develop.
5. Inclusão no próximo ciclo de release.

CRs não podem ser aplicados diretamente à main, pois não são destinados à produção imediata.

8. Processo para Correção de Bugs (Bug Fix)

Os bugs são classificados da seguinte forma:

8.1 Não Críticos

Afetam apenas o desenvolvimento ou não comprometem o produto em uso.
Fluxo:

1. Criação de uma branch bugfix/ a partir de develop.
2. Correção e testes.
3. Merge na develop.

8.2 Críticos

Afetam clientes, interrompem funcionalidades essenciais ou representam risco.
Fluxo:

1. Criação de branch hotfix/ a partir da main.
2. Correção, validação e testes acelerados.
3. Merge na main e geração de um novo PATCH (ex.: 1.2.1).
4. Merge de volta na develop para sincronização.

9. Controle Configuracional das Variantes

9.1 Artefatos Controlados

O controle inclui:

a) Código-fonte
b) Scripts de build
c) Feature toggles
d) Módulos por variante
e) Documentação
f) Configurações específicas de deploy

9.2 Feature Toggles

Permitem ativar ou desativar funcionalidades conforme variante:

- low-end: funcionalidades mínimas
- medium: funcionalidades intermediárias
- high-end: todas as funcionalidades

9.3 Módulos

Certos componentes podem existir apenas para as variantes mais completas. O build sistema detecta e compila apenas o necessário conforme perfil.

10. Liberação e Deploy

Processo de release:

1. Criação da release branch.
2. Validação e estabilização.
3. Merge na main e aplicação de tag.
4. Geração dos três artefatos (low, medium, high).
5. Registro da versão liberada.

Hotfixes seguem fluxo específico descrito na Seção 8.2.

11. Rastreabilidade

Todo item deve possuir vínculo com:

a) CR
b) Issue
c) Bug Report
d) Versão liberada
e) Branch correspondente

Ferramentas recomendadas: Git, GitLab/GitHub Issues, Jira ou Redmine.

12. Conclusão

Este Plano de CM consolida práticas formais e organizadas para suporte à linha de produto, garantindo rastreabilidade, estabilidade, padronização e qualidade. A utilização de versionamento semântico, branching estruturado e estratégias de diferenciação entre variantes assegura que o desenvolvimento e a manutenção ocorram de forma previsível e controlada.