UNICESUMAR
EVERTON CARNIATO FIRMINO DE OLIVEIRA

PLANO DE GERENCIAMENTO DA CONFIGURAÇÃO

Maringá - PR
2020

PLANO DE GERENCIAMENTO DA CONFIGURAÇÃO

Histórico do documento

Data	Versão	Descrição	Autor
20/Nov/2020	1.0	Início do plano de GC	Everton Carniato

1. Propósito

O propósito deste plano é definir o escopo do gerenciamento da configuração do produto de software, como suas ferramentas de gestão, versionamento e os processos incluídos em todo o ciclo de vida do software.

2. Ferramentas

Ferramenta	Versão	Descrição
Redmine	4.1.1	Controle do que deve ser feito (features, bugs, planejamentos, etc) e do tempo de execução.
Git	2.29.2	Versionamento.
GitHub	-	Repositório remoto do código-fonte do projeto.
Google Docs	-	Repositório dos artefatos de especificação.
Visual Studio Code	-	Editor de código-fonte
AWS	-	Cloud do produto de software

3. Controle de Funcionalidades e Desenvolvimento

O Redmine deve possuir dois projetos, um de controle das funcionalidades do produto de software e outro do controle de desenvolvimento, no controle das funcionalidades é onde ficam as features já desenvolvidas e também as que serão desenvolvidas no futuro, já o projeto de controle de desenvolvimento é onde ficam as issues das features que estão sendo desenvolvidas.

Todas as issues devem seguir um padrão para garantir uma boa rastreabilidade do que está sendo e do que já foi desenvolvido. Para isso, no Redmine, todas as issues devem possuir seu identificador com a abreviação da feature relacionada e o número sequencial, por exemplo Issue #PAG-001 relacionada a parte de pagamento da aplicação. Além disso, todas as issues devem possuir um título sucinto que indique do que se trata, no corpo da issue também é preciso conter quem é o responsável por aquela tarefa, na descrição da issue precisa ter bem especificado o contexto dessa tarefa e um DOD (Definition of Done) que é a listagem do que deve ser feito para considerar a issue como finalizada.

4. Controle de Mudança

4.1. Commits

Todo commit deve possuir mensagem de identificação seguindo o padrão a seguir:

> *#PAG-001: Exemplo de issue*
>
> *Foi desenvolvido a tela de escolha do meio de pagamento e inserção dos dados conforme a escolha.*

Onde "#PAG-001" é o identificador da tarefa no Redmine e em seguida seu título, e embaixo um texto sucinto do que foi desenvolvido no commit. Dessa forma, será mantida uma boa rastreabilidade no repositório para as features desenvolvidas.

4.2. Branchs

A estratégia de branch adotada é o Git Flow, nele existem a branch "master", a qual é versão atual do sistema e principal branch, ou seja, o código que roda o ambiente de produção. Além dela, existe a branch "develop", onde roda o ambiente de desenvolvimento/homologação e onde estão as novas features desenvolvidas que estão esperando liberação para ir para produção, e durante o desenvolvimento, cada feature deve possuir sua própria branch e quando for finalizada é feito um merge para a "develop".

4.3. Política de Merge

Para realizar o merge da branch de uma feature específica para a branch "develop" é preciso que o desenvolvedor faça um Merge Request, nele outro desenvolvedor ou o PO do projeto irá revisar o código e deve aprovar ou recusar, o mesmo processo segue para um merge da branch "develop" para a "master".