| | | | |
|---|---|---|---|
| UniCesumar | **Trabalho:**<br>Plano de gerência de configuração | **Matéria:**<br>Gerência de configuração | |
| | **Aluno:**<br>Gabriel Cilico Leite | **R.A.:**<br>1811307-2 | **Data:**<br>22/11/2020 |

# Plano de gerência de configuração

USO DO REDMINE, GIT E GOOGLE DOCS PARA
O GERENCIAMENTO DE UM PROJETO DE SOFTWARE

Maringá, 22 de novembro de 2020

| UniCesumar | **Trabalho:** Plano de gerência de configuração | **Matéria:** Gerência de configuração | |
|---|---|---|---|
| | **Aluno:** Gabriel Cilico Leite | **R.A.:** 1811307-2 | **Data:** 22/11/2020 |

Durante o processo de software, um dos maiores desafios consiste no rastreio e descrição das atividades, além do controle contínuo das alterações e novas funcionalidades para que sejam auditadas.

Para que isso seja realizado com maestria, um plano de gerenciamento de configuração é fundamental, não apenas para o processo de criação do produto, mas um plano que acompanhe toda a vida do mesmo, sendo eficaz para o relato de novas funcionalidades, ou mudanças em funcionalidades já existentes mesmo que haja mudanças nas equipes de desenvolvimento.

Neste caso, seria necessário uma ferramenta que nos permita dividir o processo de desenvolvimento entre dois ou mais desenvolvedores sem que haja interferência entre as tarefas delegadas a cada um.

O Git seria fundamental neste processo, contendo o repositório do código fonte e adotando a seguinte estratégia: teríamos uma *branch Main*, contendo a versão estável do produto, protegida pela regra de que para algo entrar na mesma, deverá passar por todos os testes automáticos, bem como todos os testes realizados pela equipe de qualidade.

Para garantir que estes testes sejam realizados, será utilizado uma *branch* intermediária, denominada por *QA*, nesta *branch* deverá entrar todas as funcionalidades que devem ser incluídas na próxima versão do produto. Sempre que receber mudanças, deverá passar por todos os testes manuais realizados pela equipe de qualidade, caso hajam falhas, a *branch* ficará em *stand-by* até que receba um merge com as correções. Caso os testes manuais sejam executados com sucesso (sem apresentar falhas), a branch receberá sinal verde para que seja reintegrada a *Main* e então se transformar em uma nova versão do sistema.

Se tratando do processo de desenvolvimento, teríamos uma *branch Development*, em que as alterações - ou novas funcionalidades - desenvolvidas, seriam *mergeadas*. Nesta, o processo se daria da seguinte forma: o desenvolvedor cria uma branch (a partir da *Development*) seguindo certos padrões de nomenclatura com o seguinte template: [tipo de tarefa]/[código da *issue*]-[descrição da funcionalidade] (como: *feature/NF-157-criando-endpoints-de-pessoa* ou *bug/CF-155-correcao-sistema-de-login)*. Assim que uma tarefa for concluída, o desenvolvedor deve abrir um *Pull Request*, que passará por *Code Review* de um ou mais desenvolvedores e caso não haja alterações, poderá ser aprovada, mas apenas será reintegrada caso todos os testes automatizados sejam executados com sucesso.

Assim que uma *Sprint* for finalizada, a *branch Developmen*t é *mergeada* para *QA*, onde passará pelos processos descritos anteriormente. Mas apenas uma estratégia de *branching* bem definida, não é suficiente para que as tarefas sejam rastreadas com eficácia, além de não garantir que as tarefas sejam devidamente priorizadas.

Agora, é expressamente necessário uma ferramenta que controle nossas *issues*, permita que sejam priorizadas e esteja integrada com o repositório. Para isso, podemos utilizar o *Redmine*. Esta ferramenta nos auxiliará em hierarquizar e rastrear as *issues*. Nela, definiremos os responsáveis pela tarefa, os solicitantes, a descrição, o tamanho, o tipo, entre outras especificações que sejam importantes no gerenciamento de mudanças.

| UniCesumar | **Trabalho:** Plano de gerência de configuração | **Matéria:** Gerência de configuração | |
|---|---|---|---|
| | **Aluno:** Gabriel Cilico Leite | **R.A.:** 1811307-2 | **Data:** 22/11/2020 |

Em cada *issue*, devemos anexar um link do *Google Docs*, onde teremos toda a especificação da tarefa, o que a motiva e o que deve mudar ou ser incluído no sistema. A *issue* no Redmine ainda contará com algumas informações importantes para o processo, como *status* da tarefa (Aberto, Em desenvolvimento, Desenvolvido, Em teste, Em reintegração, entre outros), número da *Sprint*, e o código a ser inserido na nomenclatura da *branch*.

Contudo, o processo estaria totalmente organizado, permitindo que seja identificado cada alteração, obtendo todas as informações necessárias para tomada de novas decisões, como o por que da mudança, quem modificou, quem solicitou a modificação, quando foi modificada, entre outras. Assim, o processo permitiria uma vida útil mais longa ao sistema, já que os analistas teriam em mãos todas as informações e ferramentas necessárias para apresentação rápida de soluções para o sistema.