Tarefa 7: Plano de Configuração de Software

Plano de Gestão de Configuração de Software

1.0 Introdução

O SCM apresentado tem como objetivo organizar e documentar as atividades de configuração de software da solução de organização de viagens (Software ainda não nomeado). Com isso, pretende-se manter uma padronização dos fluxos de trabalho, uso de ferramentas de controle de versão e garantir que o projeto possua um ciclo de desenvolvimento e manutenção organizado.

1.1 Escopo e objetivo das atividades de SCM

O documento de gerenciamento de configuração de software apresentado tem a intenção de detalhar as práticas de desenvolvimento utilizadas no projeto da solução. Dessa forma, todo e qualquer colaborador envolvido na criação deste software terá um manual de boas práticas a ser seguido. Esse documento organiza as informações de como modificar o software e como documentar tal mudança, fornecendo caminhos padronizados com o intuito de garantir a consistência do projeto e sua manutenção.

1.2 Papel organizacional da SCM

Considerando que o projeto está sendo desenvolvido por um grupo pequeno de desenvolvedores, todos os integrantes serão responsáveis pela realização desses papéis de gestão de configuração. Dessa forma, cada um desses desenvolvedores é responsável por seguir os padrões determinados pelo SCM e garantir que o projeto não será prejudicado por inconsistências de gerenciamento. Portanto, o indivíduo que documenta suas ações no projeto deve controlar o seu andamento e agir em harmonia com os outros integrantes da equipe.

2.0 Tarefas de SCM

Esta seção pretende detalhar as tarefas importantes de SCM.

2.1 Identificação

Todos os documentos devem seguir uma nomenclatura padrão, para que assim todos os participantes tenham o entendimento do que se trata o tal documento redigido.

Padrão da nomenclatura: NOMEDOPROJETO_ARTEFATO_DATA.

Data: O formato da data deve ser seguido do padrão estabelecido.

Padrão da data: DDMMAAAA.

Exemplo: NOME_SQAP_23052022.

Tabela de documentos com a sigla que deve ser seguida:

Artefato	SIGLA
Plano de Gerência de Projeto de Software	SPMP
Plano de Gerência de Configuração de Software	SCMP
Plano de Garantia da Qualidade de Software	SQAP
Plano de Testes do Software	STP
Pauta e Ata de Reunião	MEET
Relatórios (status, métricas, auditorias, resultado de testes)	RPT
Documentos de Arquitetura	ARCH
Artefatos Comerciais	MKT
Documentos de Requisitos	REQ
Documentos de Processo	PRC
Documentos de Testes	TST
Modelo de Classes	CMD
Modelo de Entidade-Relacionamento	ER
Caso de Uso	UC

Planilha de Estimativas	EST

2.2 Controle de Configuração

Toda e qualquer alteração na documentação do projeto\código fonte, deve partir de uma solicitação de mudança, esta deve ser analisada pelo Analista de sistemas ou o Gerente de Projeto.

Feita a solicitação, os responsáveis deverão analisar os possíveis efeitos colaterais da alteração, quais os custos e impactos que essa alteração irá causar no projeto.

Anexada junto à solicitação, um formulário deve ser preenchido, contendo todas as informações necessárias para a aprovação da alteração.

Campo	Valor
Versão	Selecionar a versão apropriada
Módulo	Descrever a camada\subprograma que irá sofrer a alteração
Prioridade	Escolher uma dentre (Baixa\Normal\Alta\Urgente)
Tipo de Problema	Escolha o tipo: Erro: relato de um defeito no artefato a ser mudado; Melhoria: melhoria do artefato/feature já existente Feature: criação de novo artefato/feature Tarefa: uma tarefa
Responsável pela solicitação	Selecionar o responsável pela CR. Em dúvida, não preencher
Responsável pela alteração	Indicar equipe que irá realizar a alteração
Resumo	Breve descrição do problema.
Descrição	Descrição detalhada do problema.

2.2.2 1) A necessidade de modificação é reconhecida.

Responsáveis: Stakeholders, Time de garantia da qualidade, Gerência do projeto

2.2.2 2) Pedido de modificação.

Responsáveis: Stakeholders, Time de garantia da qualidade, Gerência do projeto

2.2.2 3) Relatório de modificação é gerado (formulário anterior)

Responsáveis: Stakeholders, Time de garantia da qualidade, Gerência do projeto

2.2.2 4) Avaliação do pedido.

Responsáveis: Analista de sistemas, Gerente do projeto

2.2.2 5) A solicitação é colocada em fila ou o pedido é negado (interrompe aqui o processo)

Responsáveis: Analista de sistemas, Gerente do projeto.

2.2.2 6) Objetos de configuração relevantes são “reservados” (como o git será o controlador de repositório, isso significa criar uma nova branch para o desenvolvimento da alteração).

Responsáveis: Equipe de desenvolvimento.

2.2.2 7) A modificação é feita e revisada.

Responsáveis: Equipe de garantia da qualidade

2.2.2 8) Atividades de garantia de qualidade e testes são realizadas

2.2.2 9) A alteração é incorporada à versão de distribuição (merge da branch de desenvolvimento) e entrará na próxima janela de distribuição.

Responsáveis: Gerência do projeto

2.3 Controle de versões

O sistema de versionamento escolhido será o git, onde vamos criar branches e vamos fazer commits para cada alteração no sistema, dessa forma terá organização para a equipe saber o que aconteceu em cada versão e o git possibilita retornar para versões anteriores se precisar.

2.3.1 Descrição

O git trabalha com uma arquitetura de Branch, que se trata de uma ramificação, e com commit, que se trata de uma alteração no código onde o desenvolvedor consegue escrever uma breve descrição do que foi alterado no código.

1. Clone do projeto- O primeiro passo é criar um repositório em alguma plataforma de repositórios git, e clonar o projeto em sua máquina localmente.
2. Criação da Branch- Ao criar uma branch, estamos criar uma ramificação onde podemos fazer alterações no código de maneira livre sem interferir nos originais, uma boa prática quando está tratando de uma nova funcionalidade do projeto.
3. Commits- A cada alteração feita no código, o usuário deve realizar um commit salvando com uma mensagem descrevendo de forma reduzida o que foi alterado nesse commit, dessa forma fica salvo o histórico de cada alteração se um dia uma alteração quebrar o código original é só voltar o commit que essa alteração ainda não tinha sido realizada.
4. Push- Quando a funcionalidade que o desenvolvedor trabalha está concluída, deve ser enviada a branch com todas atualizações para o repositório remoto, assim ela ficará disponível para os demais poderem ver e alterar.
5. Merge- Quando finalizado a branch e subido para o repositório remoto, deve se juntar a branch com a nova funcionalidade com a branch original para que o projeto fique atualizado e completo.

2.3.1 Descrição da tarefa i de controle de versão

A cada alteração ou atualização do sistema, o versionamento segue o seguinte padrão:

<XX>.<YY>.<ZZ>

Sendo que o <XX> corresponde ao número de release principal da solução, incrementado com 01 a cada novas funcionalidades/soluções entregues. Esse incremento ocorre apenas em grandes alterações e entregas.

<YY> corresponde ao número de entrega de funcionalidade que pode ser incrementado em 01 à medida que cada nova pequena funcionalidade é entregue. Caso <XX> seja alterado, esse número inicia-se novamente com 00.

<ZZ> corresponde ao número de reparo de defeitos encontrado no sistema. Inicia-se com 00 e é incrementado com 01 a medida que uma correção de uma

funcionalidade já existente é entregue. O valor torna-se 00 toda vez que <XX> ou <YY> é alterado.

A mesma abordagem aplica-se ao versionamento da documentação através do formato <XX>.<YY>.

<XX> é a representação da última versão da documentação.

<YY> é a representação do rascunho da última versão de documentação <XX>.

Nesse formato, cada vez que uma nova versão de documentação é gerada e aprovada, o <XX> é incrementado em 01 e <YY> é zerado. Quando altera-se um documento mas ainda não aprova-se ele, incrementa-se 01 na <YY>. Dessa forma é possível manter uma certa hierarquia ao organizar as informações, possibilitando o entendimento de quais mudanças vieram antes ou depois que outras.

2.4 Comunicação do status das configurações (CSA)

As mudanças são comunicadas à equipe por documentação e reuniões em formato de planning utilizada no scrum, onde é discutido os requisitos da tarefa a ser executada. Da mesma forma que elas são revisadas na etapa de review durante a sprint.

2.4.1 Descrição

O P.O. da equipe fica responsável por validar mudanças que devem ser feitas e documentadas. Enquanto o analista fica responsável por trazer a tarefa de uma forma mais simples de entender, documentando com histórias de usuário. Os desenvolvedores da equipe ficam responsáveis pela execução das mudanças solicitadas.

2.4.2 Produtos e documentações do trabalho

Após as alterações serem concluídas, é gerado o relatório das mudanças, que traz a causa, quem realizou a demanda, quando ocorreu e o que mais será afetado por elas. Isso é feito com os *feedbacks* da equipe como um todo.

2.5 Auditorias e revisões

Para fins de auditoria das alterações, deverá ser adicionada ao final de cada etapa do projeto concluída ou alteração feita. Iremos seguir o modelo do Scrum, utilizando o momento de “revisão da sprint” para fins de auditoria. O trabalho será realizado pela equipe em conjunto. Registros e documentações serão feitos pelo documentador.

2.5.1 Descrição

Durante a auditoria, os itens que foram modificados/alterados durante o período são avaliados. Se aquilo que foi solicitado, foi entregue. Se as alterações foram feitas corretamente, eventuais melhorias técnicas são discutidas.

É checado também se todos os SCIs relacionados às modificações foram atualizados e se os procedimentos de SCM foram seguidos corretamente.

2.5.2 Produtos e documentação de trabalho

Se a auditoria encontrar correções a serem feitas nos objetos entregues, novas solicitações de alterações são criadas e registradas em uma área específica do repositório.

2.6 Coleta de dados e avaliação

Como não haverá um cliente ativo durante o desenvolvimento do projeto, os feedbacks serão internos. O P.O. e o Gerente de Projeto são encarregados de atuar como o cliente do produto, realizando avaliações do estado e apontando melhorias para o projeto. Ainda, o código passará por um responsável de QA que retornará bugs e erros a serem corrigidos.

Ao final do projeto, será liberada uma versão Beta para testes com usuários reais e então, serão coletadas informações de usabilidade e reclamações feitas por esses usuários.

3.0 Padrões, Práticas e Convenções (Standards, Practices and Conventions - SPC)

SPC que será utilizada para guiar o trabalho de SCM é descrito aqui.

Boas práticas que o time deve seguir para manter a organização em relação às mudanças que deverão ocorrer, são:

- Documentar toda alteração necessária;

- Analisar e revisar a mudança proposta;
- Dividir em tarefas a mudança;
- Comunicar à equipe qual tarefa será executada por um membro da equipe;
    - O kanban será usado para isso, com *lanes* de backlog, doing, QA e done.
- Manter o controle de versão;
    - O git será a ferramenta utilizada.
- A cada finalização de uma tarefa, ela é passada para uma revisão da equipe;
- Documentar a mudança realizada.

4.0 Recursos para SCM

Ferramenta	**Descrição**
Trello	Gestão da equipe e do projeto
ReacNative	Front-end
Java	Back-end
Oracle	SGDB a ser utilizado
GitHub	Versionamento