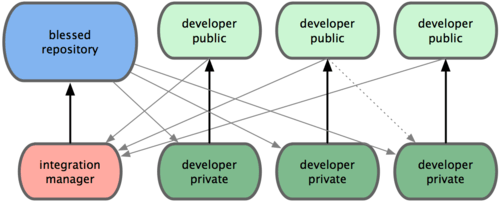
|  |  |
| --- | --- |
|  | **UNIVERSIDADE DO OESTE DE SANTA CATARINA**  **UNOESC – CHAPECÓ** |
| **Fluxos de trabalho** | |

|  |  |
| --- | --- |
| **DADOS DA DISCIPLINA:**       Disciplina: Git  Curso: Engenharia da Computação / Sistemas da Informação       Data: 23/10/17 Turno: Vespertino  Professor: Iskailer Rodrigues  Acadêmicos: Daniel Wenzel,  Jhonatan Oliveira,  Nicolas Maia,  Ayrton Lorenceti,  Lucas Medeiros   |  | | --- | |  | |
| **CONTEÚDOS:** Tipos de fluxos de trabalho em git. |

Fluxo de trabalho do Gerente de Integração

Quando iniciamos um projeto de software, e decidimos trabalhar com uma equipe de profissionais de TI, é comum a utilização do git para gerenciamento do projeto em si.

O fluxo de trabalho do gerente de integração se baseia em um gerente central que controla as informações que entram e que saem do projeto, possuindo acesso ao repositório principal (blessed repository), enquanto os demais desenvolvedores trabalham em cima de um clone do projeto privado até o momento em que é conveniente mesclar seu projeto com o principal.



Exemplo de etapas usadas pelo fluxo de trabalho gerente de integração (GitHub):

1. Gerente cria um repositório base para armazenamento do projeto principal;
2. Desenvolvedores realizam um fork do projeto para seu próprio repositório público;
3. Desenvolvedores criam um clone do projeto público para um repositório local (máquina de trabalho);
4. Desenvolvedores *criam novos arquivos/modificam arquivos* do projeto em sua máquina local;
5. Desenvolvedores realizam um pull do repositório local para seu próprio repositório público;
6. Desenvolvedores realizam um pull request de seu projeto público para dentro do repositório principal (gerente);
7. Gerente de integração avalia os arquivos do projeto e autoriza ou não o pull request;

* Caso o gerente aceite o pull request, o projeto do desenvolvedor é mesclado com o base;
* Caso o gerente negue o pull request, o projeto do desenvolvedor é ignorado (o gerente pode criar uma mensagem de resposta ao desenvolvedor, com o motivo no qual o pull foi negado);

Logo após, o processo volta à etapa 4 até a finalização do projeto.

Esse fluxo é comumente utilizado, como citado da ferramenta de repositório online GitHub, pois os desenvolvedores podem continuar trabalhando em cima do projeto local enquanto o gerente administra os pedidos de pull. E o gerente pode continuar seu trabalho enquanto os desenvolvedores progridem em seus projetos.

**Fluxo de Trabalho de Ditador e Tenentes**

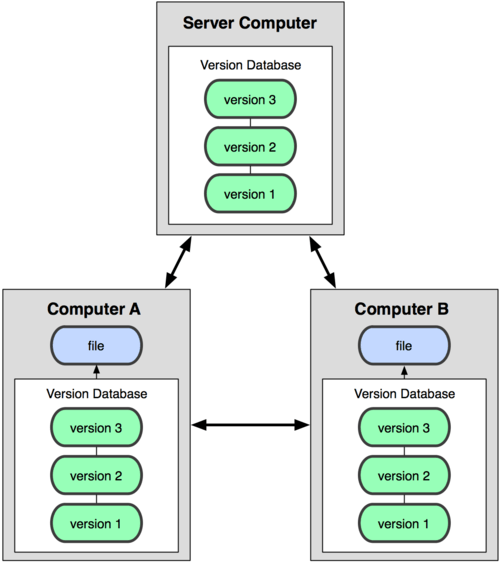
Ayrton Domingos Lorenceti

**1 Introdução**

Com o avanço dos softwares, criou-se a necessidade de registrar as mudanças feitas em um arquivo ou conjunto de arquivos ao longo do tempo. Esses registros eram guardados, substituídos e atualizados através de sistemas de controle. O Sistema de Controle de Versão Local foi o primeiro a ser utilizado, mas logo foi substituído pelo Sistema de Controle de Versão Centralizado.

Porém, esse sistema centralizado tinha uma grande falha: ele possuía um servidor central, e caso esse servidor ficasse fora do ar, os desenvolvedores não poderiam trabalhar em conjunto ou salvar novas versões de arquivo durante este período.

Para solucionar isso, surgiu o Sistema de Controle de Versão Distribuídos, na qual os clientes copiam o repositório inteiro do servidor para si, e caso o servidor principal falhe, só é preciso pegar esse repositório do cliente e copiá-lo de volta para restaurar o servidor.



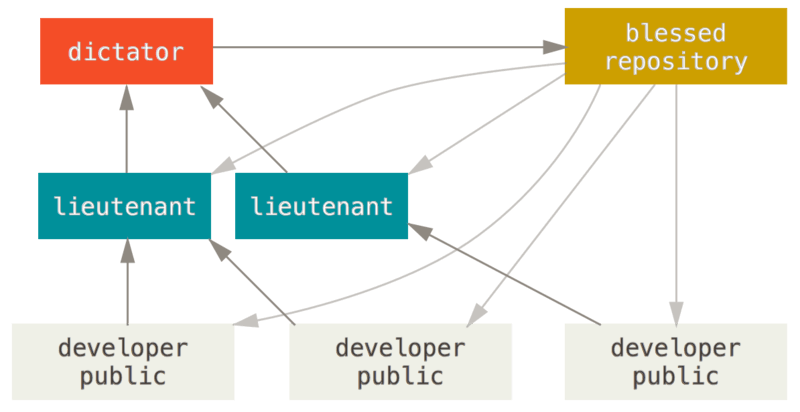
Além disso, esse sistema lida muito bem com aspecto de ter vários repositórios remotos que podem trabalhar em conjunto entre si, permitindo que se estabeleça diferentes tipos de workflow (fluxo de trabalho). O git, desenvolvido e projetado por Linus Torvalds, utiliza esse sistema.

**2 Desenvolvimento**

Um dos fluxos de trabalho que o Sistema de Controle de Versão Distribuído permite, é o Fluxo de Trabalho de Ditador e Tenente. Neste fluxo existem três papéis fundamentais: o ditador, os tenentes e os desenvolvedores.

O papel do ditador é fazer merge dos branches master de seus tenentes em seu próprio branch master, para em seguida dar push das alterações de seu branch master no repositório oficial. Já os tenentes, devem fazer merge dos topic branches dos desenvolvedores em seus branches master. Os desenvolvedores, por sua vez, trabalham em seu topic branch e baseiam seu trabalho no branch master do ditador. E, após o ditador dar push no repositório oficial, os desenvolvedores devem fazer rebase em cima desse repositório. Passo-a-passo, o processo funciona da seguinte maneira:

1. Desenvolvedores regulares trabalham em seu topic branch e baseiam seu trabalho sobre o master. O branch master é o do ditador;
2. Tenentes fazem merge dos topic branches dos desenvolvedores em seus master;
3. O ditador faz merge dos branches master dos tenentes em seu branch master;
4. O ditador dá push das alterações de seu master para o repositório de referência para que os desenvolvedores possam fazer rebase em cima dele.



**3 Conclusão**

Por possuir melhor escalabilidade, o Fluxo de Trabalho de Ditador e Tenentes é indicado para projetos grandes ou altamente hierárquicos, gerando assim menos pontos de falha. Este fluxo também permite ao ditador delegar grande parte do trabalho e recolher grandes subconjuntos do código em vários pontos antes de integrar eles.

**4 REFERÊNCIAS**

GIT-SCM. **Primeiros passos - Sobre o Controle de Versão**. Disponível em: <https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o>. Acesso em: 23 out 2017.

GIT-SCM. **Git Distribuído - Fluxos de Trabalho Distribuídos**. Disponível em: <https://git-scm.com/book/pt-br/v1/Git-Distribu%C3%ADdo-Fluxos-de-Trabalho-Distribu%C3%ADdos#Fluxo-de-Trabalho-de-Ditador-e-Tenentes>. Acesso em: 23 out 2017.

PIRES, Rafael. Desenvolvimento distribuído com Git e como aplicá-lo. **Concrete**, São Paulo, 1 mai. 2017. Disponível em: <https://www.concrete.com.br/2017/05/01/desenvolvimento-distribuido-git/>. Acesso em: 23 out 2017.

TERCEIRO, Antonio. **Controle de versão e fluxo de trabalho em projetos de desenvolvimento de software**. 2008. 171 f. Doutorado Multi-institucional em Ciências da Computação – UFBA, Salvador, 2008.