Trabalho de Pesquisa Sobre Git

Importância do Versionamento do Código Fonte (Parte 2/17)

Merges automáticos: “Mesclar” – Exemplo: Se você tem dois membros da equipe e ambos efetuam alterações no mesmo arquivo (Index.html) e efetua o Commit no arquivo sendo feito a atualização do arquivo, quando o outro membro também efetuar a atualização no mesmo arquivo (Index.html) e Git tem uma capacidade de inteligência de conseguir “Mesclar” ambas as atualizações em um único arquivo, se em caso de ambos tiverem feitos alteração no mesmo bloco do arquivo haverá um conflito, agora se forem em locais diferentes o Git tem a capacidade de “Mesclar” esses arquivos.

Permite retornar uma versão anterior do código com facilidade, se você perceber que há alguma alteração que você não consiga corrigir ou acabou perdendo o arquivo novo que estava alterando e não deu tempo de dar o Commit, pode fazer o download da versão anterior e começar de novo.

Permite integração com servidores de integração continua utilizado para automatizar o processo de deploy da aplicação.

Criando uma conta no Github (Parte 5/17)

GitHub é um servidor de hospedagem para git, é gratuito para repositórios públicos (projetos open Source), ele é muito bom para hospedagem de códigos pessoais ou projetos open source, ótimo para manter um projeto colaborativo e manter códigos criados por você no GitHub é ótimo como referencia no CV

Pagina Oficial: https://github.com

Criando um Repositório no Github (Parte 6/17)

1. Clique no Botão “New repository”
2. Digite “tutorial-git” no campo “Repository name”
3. Selecione “Initialize this repository with Readme” para criar um arquivo descritivo do projeto (opcional)
4. Clique no botão “Add .gitignore” e selecione “Node” para inicializar o projeto com a informação de arquivo e diretórios a serem ignorados no commit.
5. Clique no botão “Create repository”

O que é README: Ele é um arquivo com contem a descrição do seu projeto, o Github utiliza um arquivo no formato de marcação, ele tem uns caracteres especiais para você fazer uma marcação nele ele permite a criação nele, fazendo com que você possa criar uma descrição do seu projeto, com instruções de como instalar, isso é muito bom para quem não conhece o projeto ou até mesmo entre membros da equipe.

O que é .gitignore: É um arquivo que só vai conter umas entradas dos arquivos ou diretórios que ele não deve enviar para o servidor remoto.

Comando git init (Parte 7/17)

* O comando ‘git init’ serve para inicializar um diretório vazio com as metadadas de um projeto Git
* Um diretório oculto chamado ‘.git’ armazena as informações do projeto
* É o primeiro comando a ser digitado ao se criar um projeto Git.

Comando git status (Parte 8/17)

* O comando ‘git status’ serve para informar os detalhes do repositório
* Dentre os detalhes são exibidos novos arquivos, arquivos modificados ou excluídos.
* Também são exibidos informações sobre arquivos pendentes de commit
* Sempre digite o comando de status antes de qualquer ação para certificar o status atual de seu repositório

branch master: O código principal que vai para a produção

Comando git remote (Parte 9/17)

* O comando ‘git remote’ serve para associar um repositório remoto com nosso projeto local
* Utilizamos a opção ‘add origin URL’ para informar o repositório a qual a associação será realizada

Comando git pull (Parte 10/17)

* O comando ‘git pull’ serve para baixar os arquivos atualizados do repositório remoto
* Ele deve ser utilizado em conjunto com a origem desejada, que em nosso caso será ‘origin master’.

+ ‘master’ é branch principal onde o código ficará armazenado.

Comando git add (Parte 11/17)

* O comando ‘git add’ serve para adicionar um ou mais arquivos à uma fila do Git para que o commit seja realizado posteriormente.
* Ele aceita um ou mais arquivos como parâmetro, separados por espaço.
* Caracteres especiais como ‘\*’ podem ser utilizados para adicionar mais de um arquivo por nome ou extensão, como ‘\*.html’, ‘arquivo\*.txt’

Comando git commit (Parte 12/17)

* O comando ‘git commit serve para adicionar o código previamente adicionado pra um repositório local que o Git mantem antes do código ser enviado para o servidor
* Uma nova revisão e log são criados, porem o código continua somente sendo visível localmente
* É o ultimo passo antes do envio ao servidor remoto
* Uma mensagem com descrição da alteração é obrigatória, e deve ser informada com a utilização do parâmetro ‘-m’.

Comando git push (Parte 13/17)

* O comando ‘git push’ serve para efetivamente enviar nossas alterações para o repositório remoto
* Ao executar o comando é necessário informar a origem com o parâmetro ‘origin master’, onde ‘master’ é a branch padrão do Git.

Comando git log (Parte 14/17)

* O comando ‘git log’ serve para exibir as revisões e informações dos commits realizados no repositório
* Essa informação é útil para verificar a revisão atual, assim como pode ser utilizada para voltar o código a uma revisão anterior.

Comando git diff (Parte 15/17)

* O comando ‘git diff’ serve para comparar um arquivo em repositório ou revisões distintas
* Ele necessita de um parâmetro informando a revisão a ser comparada
* Utilize ‘HEAD’ como parâmetro para comparar as alterações com a ultima revisão disponíveis.

GitHub Desktop (Parte 16/17)

* O GitHub Desktop é uma aplicação de com interface gráfica para gerenciamento de projetos GitHub
* Ela esta disponível para Windows e Mac, e pode ser baixada em <https://desktop.github.com>
* Ela executa todos os comandos aprendidos, porem de modo automático.
* A interface facilita a visualização do projeto, principalmente no que se refere a alterações realizadas.

TBD – Trunk Based Development (Parte 17/17)

* O Git mantem uma branch principal chamada ‘master’, que contem o código final do repositório.
* Ele permite a criação de novas branches, implementação, que serão mescladas com a branch ‘master’ assim que finalizadas.

Segue a baixo os detalhes sobre o TBD:

1. O TBD seria o desenvolvimento somente no Trunk (branch ‘master’)
2. Ele evita perder tempo com criação e merges de branches
3. Permite manter sempre o código atualizado
4. É uma pratica recomendada para automatizar processos de deploy
5. É difícil de ser implementado porque a equipe deve ser muito confidente em relação ao código criado
6. Em casos específicos a criação de branches auxiliares é permitida.