**Faculdade de Tecnologia Dr. Thomaz Novelino**

**Disciplina Eletiva – Programação Web**

**Prof. Me. Fausto G. Cintra  
Relatório Acadêmico**

**Git**

**Iury Almeida de Oliveira**

**Socrates cheiragato**

1. O que é versionamento de código?

Socrates

2.Quais as vantagens da utilização do versionamento de código?

Socrates

**3.O que é o Git?**

Git trata-se de um sistema a qual gerencia todo o processo de desenvolvimento de arquivos, ou seja, cria versões a cada modificação feita nos mesmos. Permite também que várias pessoas contribuam para um projeto simultaneamente.

Git trata-se de um sistema de controle de versão de arquivos. Através deles podemos desenvolver projetos na qual diversas pessoas podem contribuir simultaneamente no mesmo, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.

Se não houver um sistema de versão, imagine o caos entre duas pessoas abrindo o mesmo arquivo ao mesmo tempo. Uma das aplicações do git é justamente essa, permitir que um arquivo possa ser editado ao mesmo tempo por pessoas diferentes. Por mais complexo que isso seja, ele tenta manter tudo em ordem para evitar problemas para nós desenvolvedores.

4. Qual o fluxo de trabalho e os principais comandos utilizados no git durante o

processo de versionamento de código, especialmente nas tarefas de:

4.1. inicialização de repositório

Para inicializar um repositório na sua máquina local, você pode começar com um novo projeto, ou um projeto já existente. Para isso você deve navegar até o diretório do seu projeto, e executar o comando:

$ git init

4.2. Clonagem de um repositorio já existente

Para obter os arquivos de um projeto em um servidor remoto, você deve navegar até o diretório a qual deseja obter os arquivos, e então executar o comando:

$ git clone URL do servidor remoto

4.3. *Commit* de código da estação local para o repositório remoto

Socrates

4.4. Como atualizar um repositório local com o código mais recente do

repositório remoto

socrates

5. O que é e como funcionam as tarefas de:

5.1. Fork de repositório

Quando um desenvolvedor deseja contribuir com o projeto de outrem, então é executado o “fork de repositório”. O “Fork” copia o repositório do dono para o repositório do contribuinte, assim possibilitando que o contribuinte realize quaisquer alterações e/ou inserções no projeto. O contribuinte pode fazer commits e push’s normalmente, e ao término solicitar ao dono que implemente tais alterações e/ou inserções.

iury

5.2. Criação de manutenção de Branches

As Branches são como uma “cópia espelho” do projeto, por exemplo, um desenvolvedor está trabalhando em um projeto sobre a branche máster (geralmente é o padrão) e outro desenvolvedor deseja contribuir para esse projeto e deseja incluir novos arquivos, novas funcionalidades, mas não deseja trata-lo como oficial ainda. Neste caso, o contribuinte efetua o fork e a clonagem do projeto original, e então cria uma nova “Branche” (alternativa) para que possa efetuar quaisquer alterações e/ou inserções que desejar sem comprometer o projeto original. Ao término do trabalho, o desenvolvedor contribuinte efetua um “Merge” para incluir tudo o que ele desenvolveu na “Branche” alternativa à “Branche” master, e então solicita um “pull request” para o dono do projeto original. O dono do projeto analisa o que foi feito, e decide incluir ou não tais alterações e/ou inserções.

Outro fator importante do git (e essa é um dos seus diferenciais em relação ao svn – caso vc o conheça) é a possibilidade de criar, a qualquer momento, vários snapshots do seu projeto, ou como chamamos mais “nerdmenete”, branch. Suponha que o seu projeto seja um site html, e você deseja criar uma nova seção no seu código HTML, mas naquele momento você não deseja que estas alterações estejam disponíveis para mais ninguém, só para você. Isso é, você quer alterar o projeto (incluindo vários arquivos nele), mas ainda não quer que isso seja tratado como “oficial” para outras pessoas, então vc cria um branch (como se fosse uma cópia espelho) e então trabalha apenas nesse branch, até acertar todos os detalhes dele. Após isso, você pode fazer um merge de volta do seu branch até o projeto original. Veja bem, se tudo isso que você leu só ajudou a te confundir mais – respire fundo – e siga em frente. Com exemplos tudo fica melhor.

iury

5.3. Pull request

socrates