**FACULDADE DE TECNOLOGIA SENAC GOIÁS**

**Gestão da Tecnologia da Informação**

LogoSenac

Denise Diniz A. Benevides

**Gerência de Configuração**

Elias Ferreira

GOIÂNIA,

2016Denise Diniz A. Benevides

**Gerência de Configuração**

Relatório apresentado como requisito parcial para obtenção de nota para a aprovação na disciplina Auditoria e qualidade de Software no Curso de Gestão da Tecnologia da Informação, na Faculdade de Tecnologia Senac Goiás.

Elias Ferreira

GOIÂNIA,

2016

**RESUMO**

Os sistemas de software estão em constante evolução. A manutenção do software, isto é, modificações em artefatos existentes, chega a consumir 75% do custo total do seu ciclo de vida. Aproximadamente, 20% de todo o esforço de manutenção é usado para consertar erros de implementação e os outros 80% são utilizados na adaptação do software em função de modificações em requisitos funcionais, regras de negócios e na reengenharia da aplicação. A Gerência de Configuração de Software surgiu da necessidade de controlar estas modificações, por meio de métodos e ferramentas, com o intuito de maximizar a produtividade e minimizar os erros cometidos durante a evolução. É uma disciplina que controlam e notificam as inúmeras correções, extensões e adaptações aplicadas durante o ciclo de vida do software de forma a assegurar um processo de desenvolvimento e evolução sistemático e rastreável, sendo indispensável quando equipes manipulam, muitas vezes em conjunto, artefatos comuns. Apesar de existir um forte apelo para o uso da Gerência de Configuração de Software durante a etapa de manutenção, a sua aplicação não se restringe somente a essa etapa do ciclo de vida do software. O uso dos sistemas de Gerência de Configuração é fundamental para prover controle sobre os artefatos produzidos e modificados por diferentes recursos desde o planejamento e levantamento de requisitos até a construção e entrega do produto.

**SUMÁRIO**

RESUMO 3

1 INTRODUÇÃO 5

2 OBJETIVO 5

3 DESENVOLVIMENTO 6

5 CONCLUSÃO 19

6 REFERÊNCIAS BIBLIOGRÁFICAS 20

# INTRODUÇÃO

As mudanças são inevitáveis quando um software de computador é construído. E as mudanças aumentam o nível de confusão entre os engenheiros de software que estão trabalhando num projeto. A confusão surge quando as mudanças não são analisadas antes de serem feitas, registradas antes de serem implementadas, relatadas aos que precisam tomar conhecimento delas e controladas de um modo que melhore a qualidade e reduza o erro. (PRESSMAN, 2007).

A mudança é um fato na vida do desenvolvimento de software. Clientes querem modificar requisitos. Desenvolvedores querem modificar a abordagem técnica. Administração quer modificar a abordagem ao projeto. Por que todas essas modificações? A resposta é realmente bastante simples. Á medida que o tempo passa todo o envolvido sabe mais sobre o que precisa, e como colocá-la em prática.

É importante fazer uma distinção clara entre manutenção de software e gerenciamento de configuração. A manutenção é um conjunto de atividades de engenharia de software que acontece depois que o software é entregue ao cliente e posto em operação. O gerenciamento de configuração de software é um conjunto de atividades de controle e rastreamento que começa quando um projeto de desenvolvimento de software se inicia e termina somente quando o software é tirado de operação.

Uma meta primordial da metodologia de engenharia de software é melhorar a facilidade com que as mudanças podem ser acomodadas e reduzir a quantidade de esforço despendido quando as mudanças são feita

A meta deste trabalho é apresentar a Gerência de Configuração no contexto da engenharia de software, a importância no controle da evolução de sistemas de software, destacar a influência da Gerência de Configuração na qualidade de sistemas de software. Tem como objetivo definir as atividades e procedimentos da gerência de configuração, de forma a permitir que se tenha um maior controle das mudanças que venham a ocorrer nos projetos durante todo o processo, inclusive da mudança de requisitos.

# DESENVOLVIMENTO

**Gerência de Configuração**

O gerenciamento de configuração é o desenvolvimento e o uso de padrões e procedimentos para o gerenciamento de sistemas de software em desenvolvimento. O gerenciamento de configuração de software (Software Configuration Management – SCM) é uma atividade abrangente que é aplicada em todo o processo de engenharia de software. Uma vez que uma mudança pode ocorrer a qualquer tempo, as atividades de SCM são desenvolvidas para (1) identificar a mudança; (2) controla a mudança; (3) garantir que a mudança esteja adequadamente implementada; e (4) relatar a mudança a outras pessoas que possam ter interesse nela.

Os procedimentos de gerenciamento de configuração definem como registrar e processar mudanças de sistema e os métodos usados para identificar diferentes versões dele. Ferramentas de gerenciamento de configuração são usadas para armazenar versões de componentes do sistema, sistemas construídos com base nesses componentes e rastrear os releases das versões do sistema para os clientes.

O gerenciamento de configuração é considerado algumas vezes parte de gerenciamento da qualidade de software, tendo o mesmo gerente dividindo as responsabilidades do gerenciamento de qualidade e de configuração. O software é inicialmente liberado pela equipe desenvolvimento para a garantia da qualidade. A equipe de GQA verifica se o sistema possui qualidade aceitável. Dessa maneira, ele se transforma num sistema controlado, ou seja, as mudanças que sofrer devem ser acordadas e registradas antes de serem implementadas. Os sistemas controlados são, algumas vezes, chamados de baseline, porque são considerados os pontos de partida para futuras evoluções controladas do sistema.

Há muitas razões por que os sistemas de existem em diferentes configurações. Configurações podem ser produzidas para diferentes computadores, para diferentes sistemas operacionais, incorporando funções específicas de clientes. Os gerentes de configuração são responsáveis por manter a rastreabilidade das diferenças entre versões de software, para assegurar que as novas versões sejam derivadas de maneira controlada e liberar novas versões para clientes certos no momento certo.

A definição e o uso de padrões de gerenciamento de configuração são essenciais para a certificação da qualidade tanto para o padrão ISO 9000 quanto para os padrões CMM e CMMI (Paul ET AL. 1995; Ahern ET AL. 2002; Peach, 1996). Em uma empresa, esses padrões devem ser incorporados no manual da qualidade ou no guia de gerenciamento de configuração. Naturalmente padrões genéricos externos podem ser usados como referência para padrões organizacionais detalhados que configuram um ambiente específico.

Em um processo tradicional de desenvolvimento de software baseado no modelo cascata, o software é entregue para a equipe de gerenciamento de configuração depois que o desenvolvimento está completo e os componentes individuais de software foram testados. Essa equipe, portanto, assume a responsabilidade pela construção do sistema completo e pelo gerenciamento de testes de sistema. Defeitos descobertos durante os testes são encaminhados á equipe de desenvolvimento para serem reparados. Depois de reparados, a equipe de desenvolvimento entrega uma nova versão do componente reparado para a equipe de qualidade. Se a qualidade for aceitável, ele pode se tornar uma nova baseline para os desenvolvimentos futuros do sistema. Baseline é um conceito de gerenciamento de configuração de software que nos ajuda a controlar as mudanças, sem impedir seriamente as mudanças justificáveis.

Em geral a GCS é aplicada apenas quando existe um processo de desenvolvimento bem definido, com atividades agrupadas em fases, constituídas por objetivos bem definidos e documentados. Neste contexto, GCS atua como um suporte sobre o qual as fases do desenvolvimento são conduzidas e os produtos controlados.

**Itens de configuração**

Em um grande sistema de software, pode haver módulos de milhares de códigos-fonte, scripts de testes, documentos de projeto etc. Eles são produzidos por pessoas diferentes e, quando criados, podem ser denominados com nomes similares ou idênticos. Para manter a rastreabilidade de todas as informações de maneira que o arquivo certo possa ser encontrado quando for necessário, você necessita de um esquema de identificação consistente para todos os itens no sistema de gerenciamento de configuração.

Entende-se como item de configuração “Cada um dos elementos de informação que são criados durante o desenvolvimento de um produto de software, ou que para este desenvolvimento sejam necessários, que são identificados de maneira única e cuja evolução é passível de rastreamento” (Pressman em [PRE 92]).

Nesta definição, tanto os documentos como os arquivos-fonte que compõem um produto de software são Itens de Configuração (IC), assim como também o são as ferramentas de software necessárias para o desenvolvimento. Se o item de configuração for composto exclusivamente de software, ele é então caracterizado como Item de Configuração de Software (ICSW).

Qualquer IC constitui uma unidade funcional que possui um ciclo de desenvolvimento e acompanhamento de GCS próprios. Qualquer sistema em desenvolvimento deve ser  particionado em itens de configuração, e o seu desenvolvimento é visto como o desenvolvimento de cada um dos ICs. Cada IC, por sua vez pode ser particionado em outro conjunto de ICs e assim sucessivamente, até que se resulte um conjunto de ICs não particionáveis, que são desenvolvidos segundo um ciclo de vida propriamente definido. A configuração de um sistema de software passa a ser definida pela configuração do conjunto dos ICSW que o constitui.

Em cada fase do processo de desenvolvimento, um conjunto bem definido de itens de configuração deve ser estabelecido. Tal conjunto representa um estágio do desenvolvimento, o qual é passível de modificações apenas mediante um mecanismo formal de alterações. A este conjunto é dado o nome de Baselines, ou Configurações Base do sistema.



Figura 1. Exemplos de itens de configuração

O desenvolvimento com configurações base pode, então, ser resumido nos seguintes pontos:

* Caracterização do ciclo de vida, identificando-se as fases pelas quais o desenvolvimento do software irá passar e, dentro delas, as atividades a serem realizadas e os produtos a serem desenvolvidos.
* Definição do conjunto de baselines. Para cada baseline planejada, deve se estabelecer quais serão os ICs que a irão compor e quais as condições impostas para seu estabelecimento;
* Baselines representam marcos no processo de desenvolvimento: uma nova baseline é estabelecida no final de cada fase do ciclo de vida do software;
* Durante cada fase, o desenvolvimento dos ICs a ela referentes está sob total controle de seus desenvolvedores, e realiza-se com ampla liberdade, podendo os ICs ser criados e modificados com bastante facilidade;
* Durante cada fase, entretanto, a modificação de uma configuração base anteriormente estabelecida somente pode ser feita de forma controlada, mediante um processo bem definido;
* Ao ser estabelecida cada baseline incorpora integralmente a anterior. Desta forma, em qualquer instante do desenvolvimento, a última baseline estabelecida representa o estado atual do desenvolvimento como um todo;
* O estabelecimento de cada baseline somente é realizado após ser aprovada por procedimentos de consistência interna, verificação e validação.

Desta forma é possível ter um controle sistemático sobre todos os itens de configuração abordados em cada fase do ciclo de vida do software, assim como é possível avaliar mais facilmente o seu grau de evolução.

**Resultados Esperados da Gerência de Configuração MSP. BR**

O propósito do processo Gerência de Configuração é estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos. Resultados esperados:

**GCO 1.** Um Sistema de Gerência de Configuração é estabelecido e mantido;

**GCO 2.** Os itens de configuração são identificados com base em critérios estabelecidos;

**GCO 3.** Os itens de configuração sujeitos a um controle formal são colocados sob baseline;

**GCO 4.** A situação dos itens de configuração e das baselines é registrada ao longo do tempo e disponibilizada;

**GCO 5.** Modificações em itens de configuração são controladas;

**GCO 6.** O armazenamento, o manuseio e a liberação de itens de configuração e baselines são controlados;

**GCO 7.** Auditorias de configuração são realizadas objetivamente para assegurar que as baselines e os itens de configuração estejam íntegros, completos e consistentes.

**Atividades de GCO**

Atividades de controle de configuração incluem a requisição, avaliação, aprovação e implementação de mudanças em itens de configuração. Mudanças podem incluir tanto correções de erros quanto melhoramentos. O grau de formalidade necessário para o processo de controle de mudança depende da linha base afetada e do impacto da mudança na estrutura da configuração.

No decorrer dos projetos, serão geradas revisões na ferramenta Subversion. Nestas revisões serão armazenadas as atualizações no cronograma, as atualizações do demonstrativo de coleta de métricas, os registros de reuniões produzidos e os documentos alterados que a gerência do projeto julgue importante sua inclusão na revisão.

Todos os documentos incluídos nas revisões são revistos pela gerência de configuração antes de sua inclusão na revisão. Pendências encontradas durante a geração destas revisões deverão ser anotadas e poderão ser ajustadas posteriormente, desde que, com o consentimento do gerente do projeto, visto que os artefatos já foram revisados anteriormente pela qualidade no decorrer das atividades.

Ao final de cada fase do projeto e das iterações serão geradas revisões que serão consideradas configurações-base (baseline) do projeto. Para estas revisões todos os documentos incluídos deverão estar revisados e aprovados pelos envolvidos.

A seguir apresentarei uma ferramenta que nos auxilia para o controle das mudanças, ou seja, controla a versão de toda alteração ocorrida no processo de software.

**GitHub**

Em poucas palavras, o Git é um sistema de controle de versão (SCM) distribuído, criado em 2005 por Linus Torvalds, o criador do Linux. Seu grande diferencial frente às opções existentes (CVS, Subversion, Clearcase etc.) é a última palavra do grifo acima: distribuído. Com ele abre-se mão da necessidade de um servidor remoto que controle o versionamento de arquivos. Cada instalação do Git é seu próprio servidor, que pode interagir com um sem-número de servidores remotos.

O GitHub, por sua vez, é um repositório de software que cresceu a ponto de tornar-se uma rede social de desenvolvimento que, dentre suas muitas funcionalidades, possibilita o colaborativismo, ou seja, permitir que qualquer desenvolvedor contribua com projetos open-source. Tudo usando Git, claro.

Outro fator importante do git (e essa é um dos seus diferenciais em relação ao svn – caso você 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. Onde altera o projeto, incluindo vários arquivos nele, mas ainda não quer que isso seja tratado como oficial para outras pessoas, então cria um branch como se fosse uma cópia espelho e então trabalha apenas nesse branch, até acertar todos os detalhes do site.

O controle de versão é um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo, de forma que você possa recuperar versões específicas. Se quiser manter todas as versões de uma imagem ou layout, usar um Sistema de Controle de Versão (Version Control System ou VCS) é uma decisão sábia. Ele permite reverter arquivos e projetos inteiros para um estado anterior, compara mudanças feitas ao decorrer do tempo, vê quem foi o último a modificar algo que pode estar causando problemas, quem introduziu um bug e quando, e muito mais. Usar um VCS normalmente significa que caso tenha estragado ou perdido algum arquivo, poderá facilmente reavê-los. Além disso, você pode controlar tudo sem maiores esforços.

**Instalando GitHub**

O GiHubt é um programa que pode ser instalado para Windows,  Mac, ou então se você usa uma VM na nuvem, como o cloud9 ou koding, o git já estará disponível em sua linha de comando.através do comando sudo apt-get install git para plataformas Linux/Debian, como o Ubuntu.

A instalação da ferramenta deve ser feita de acordo com os requisitos do sistema operacional do seu hardware, por isso, deve observar a versão compatível com o sistema utilizado. Após fazer a escolha, clica em download que imediatamente iniciará a instalação.

Imagem abaixo mostra a página para download da ferramenta:

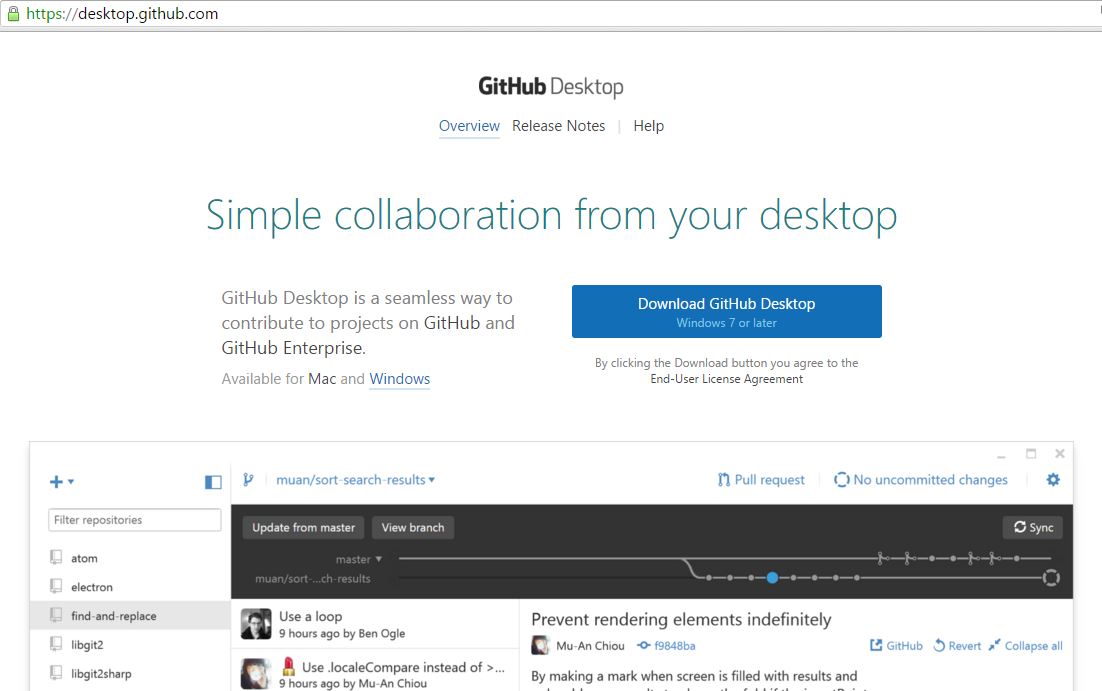


Figura 2. Imagem da página

Logo após o download, aparecerá uma mensagem na tela para a confirmação da instalação.

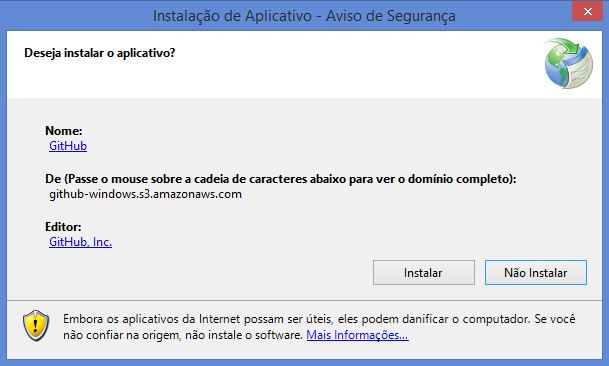


Figura 3. Imagem da mensagem de confirmação

Imagem abaixo mostra a instalação confirmada e executando:

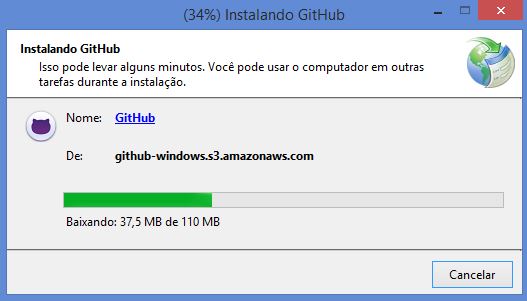


Figura 4. Imagem da execução da instalação

Instalação concluída abre a tela de boas vindas para criação da conta. Adiciona o login e senha que serão exigidos para autentificação do usuário.

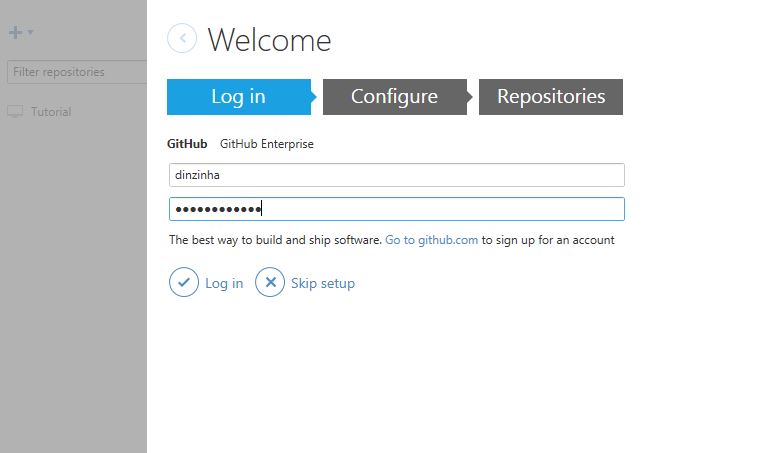


Figura 5. Imagem da criação da conta

As configurações, como foto no perfil, nome e email para segurança e notificações são realizadas em Configure. Depois de preencher todos os dados necessários, clica em Continue.

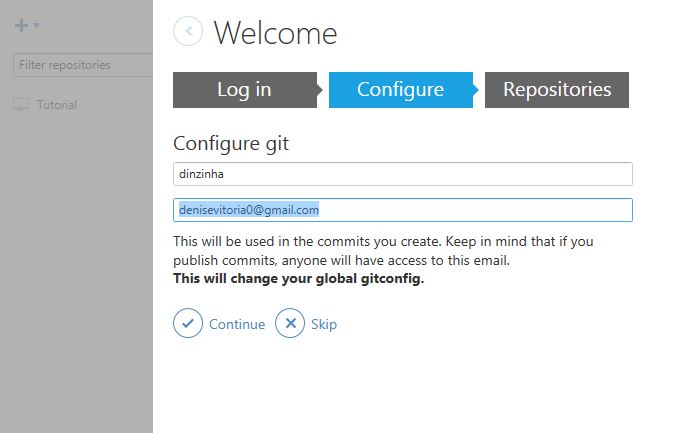


Figura 6. Imagem da configuração da conta

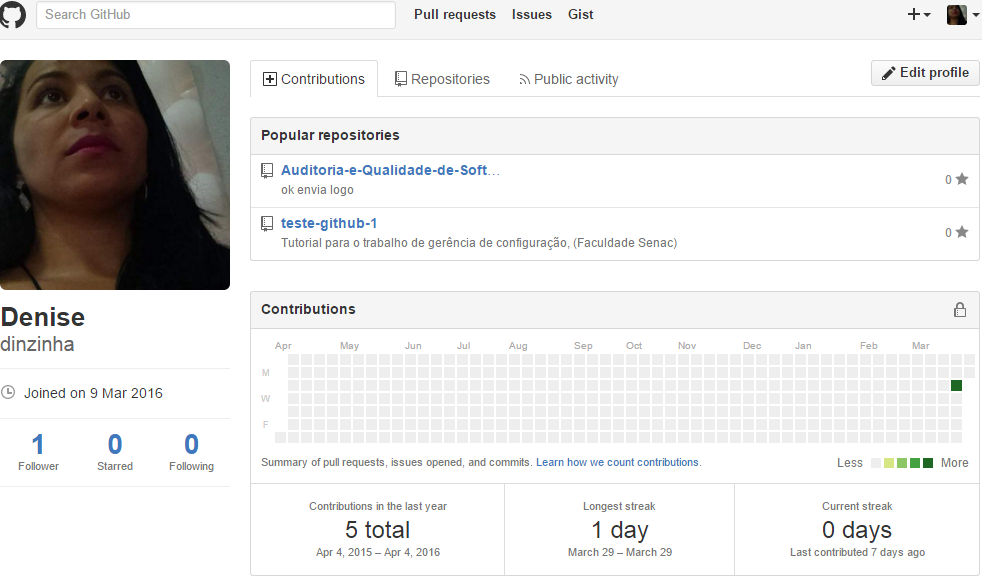


Figura 7. Imagem de um perfil configurado

Nessa etapa, já pode ser criado um repositório onde controla toda a alteração efetuada de processos, projetos de negócio e muitos outros.

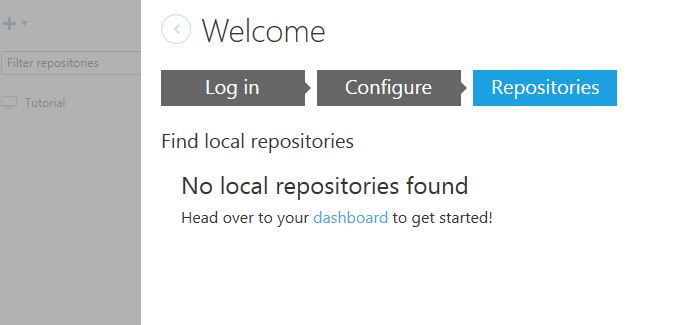


Figura 8. Imagem do Aplicativo todo configurado

Nesta imagem estamos criando um repositório cujo nome é Gerência de Configuração, de domínio público (podem ser criados reps privados pagando uma mensalidade), e com o arquivo  README. md embutido, que contém uma descrição do seu projeto. Para que possamos começar a entender como o git funciona, é fundamental criar um rep como este para os nossos testes.

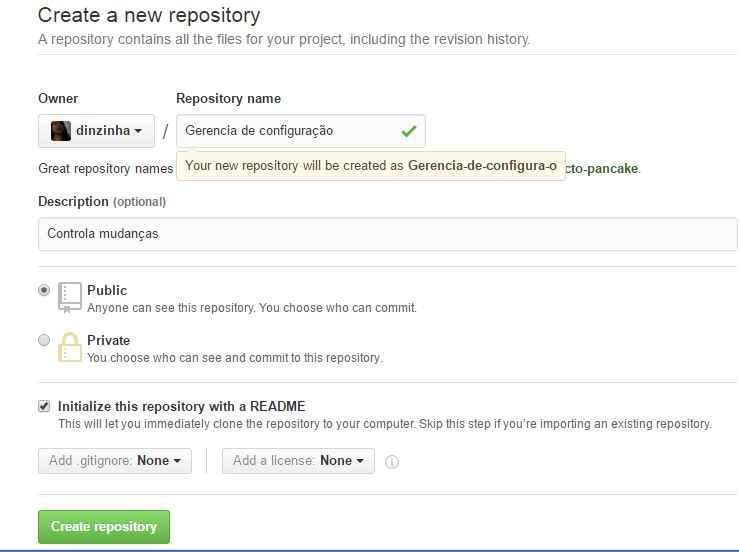


Figura 9. Criando repositório

Após a criação do repositório, ele estará disponível no endereço eletrônico <https: //github.com/<dinzinha>/Gerencia-de-configuração>, onde username é o login que você usou para se cadastrar.

Dada a introdução, a partir deste ponto, pressuponho que você tenha os conhecimentos básicos sobre Git e GitHub, ou seja, o Git está instalado em sua máquina, você já sabe como utilizá-lo basicamente (usando a linha de comando e não algum cliente visual) e você já possui uma conta no GitHub. Tudo pronto, basta seguir os passos abaixo:

**1. Fork**

Ao acessar uma página de um projeto no GitHub, basta clicar no botão “Fork”, localizado no canto direito superior. Este comando fará uma cópia do projeto em sua conta no GitHub. A partir deste momento, qualquer alteração que você submeter ao GitHub será enviada apenas para seu projeto.

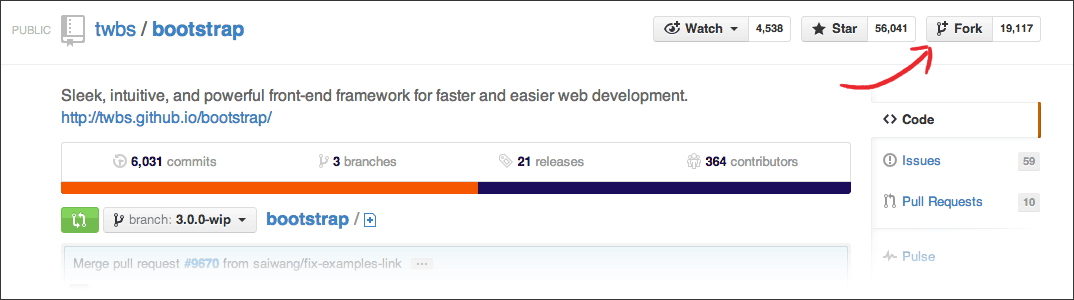


Figura 10. Imagem Fork

**2. Clone**

Após possuir uma cópia em sua conta, você precisa baixar o projeto para sua máquina (clonar) a fim abrir os arquivos em seu editor favorito. O comando a seguir mostra como fazer este clone:

**$ git clone clone-url**

O clone-url é a URL informada pelo GitHub na página do projeto, localizada na coluna da direta, sob o título “XYZ clone URL”, sendo XYZ a referência ao método de importação. Atualmente existem as opções HTTS, SSH, Subversion. Para mais informações, acesse [este link](https://help.github.com/articles/which-remote-url-should-i-use).



Figura 11. Imagem dos repositórios existentes

**3. Branc**

Quando um projeto é clonado, você receberá a versão mais atual do branch principal, chamado de master. Um conceito geral do git é o trabalho constante com topic branches, também conhecidos como feature branches. Ou seja, pelo senso comum, você nunca irá trabalhar na versão master de um projeto. Você deverá criar um branch de acordo com o tópico ou a funcionalidade que irá executar.

Por quê?

Desta forma, você terá controle do desenvolvimento e, no caso da necessidade da correção de código ou retorno a versões específicas, você saberá que aquele branch em questão refere-se apenas àquilo que você se propôs a fazer. Para tal, executa-se o comando:

**$ git co -b nome-do-branch**

Sendo co a abreviação de checkout e -b informa que quero criar um branch. Em nosso exemplo, vamos supor que eu queira atualizar a nomenclatura de classes CSS do projeto. Desta forma:

**$ git co -b new-css-naming-rules**

Pronto, agora você já estará desenvolvendo no branch criado e o master estará são e salvo.

**4. Commit**

Após fazer as contribuições desejadas, chegou a hora de registrá-las no git (lembre-se que escrevi git e não GitHub). Para isto:

**$ git commit -am "Updates the css naming rules"**

-a representa que você quer registrar todos (all) os arquivos alterados e m representa que você deseja inserir um comentário, que vem logo em seguida, entre as aspas duplas (Inicie o comentário na terceira pessoa do singular, pois ele representa a ação do commit). Lembre-se de sempre escrever no idioma do projeto para fazer comentários. Um projeto open-source bem organizado sempre utilizará o idioma inglês em sua totalidade (variáveis, métodos, comentários etc.), uma vez que, sendo o Inglês o esperanto da programação, nenhum desenvolvedor gostaria de restringir contribuições apenas de programadores que falassem apenas seu idioma.

**5. Push**

Agora que o método commit foi executado, o código estará disponível no GitHub, certo? Errado. Se você fez a lição de casa e entendeu como o git funciona, sabe que o commit apenas registrou o envio de nova versão. Agora é necessário fazer o upload das alterações para o GitHub e informar a comunidade sobre sua contribuição. Para isso:

**$ git push origin nome-do-branch**

O comando push avisa ao git que você deseja enviar sua alteração para o destino original (origin), seguido do branch que deseja enviar. Aí você, curioso que é, deseja saber como o git sabe o que significa origin. Ele mantém um registro dos repositórios existentes. Se você desejar saber, basta executar o comando git remote.

**6. Pull Request**

Até este momento, você enviou suas alterações para o branch criado em sua conta no GitHub. Agora chegou a hora de avisar o pessoal do Twitter Bootstrap que você tem algo bem interessante para compartilhar. Para tal, ao acessar a página do projeto em sua conta no GitHub, ele automaticamente mostrará o branch que você enviou e mostrará o botão “Pull Request”.

Basta informar na caixa de comentários um detalhamento do quê você fez e porque você fez. O responsável pelo repositório agora poderá simplesmente aceitar sua sugestão, dialogar com você tirando dúvidas ou solicitando alterações ou até mesmo negar sua proposta. E toda a comunidade do GitHub poderá comentar o que você fez.

**7. Sincronizando com repositórios remotos**

Você deverá, primeiramente, adicionar a referência ao repositório remoto:

**$ git remote add nome\_repositorio\_remoto** [**https://github.com/octocat/repo.git**](https://github.com/octocat/repo.git)

Agora, quer saber quais são os repositórios remotos existentes?

**$ git remote –v**

Em seguida, para atualizar o seu conteúdo local com a versão existente no repositório remoto:

**$ get fetch nome\_repositorio\_remoto master**

Para sobrescrever todo seu conteúdo local com o conteúdo do repositório remoto:

**$ git rebase nome\_repositorio\_remoto/master**

—

São seis simples passos a serem seguidos e, uma vez executados, tornam-se automáticos e extremamente prazerosos. Na comunidade de desenvolvimento de projetos abertos e até empresas de TI com mentalidade mais contemporânea valorizam muito mais o compartilhamento de conhecimento do que currículos extensos e intermináveis certificações.

Um usuário no GitHub com projetos compartilhados e um perfil social com diversas contribuições e participação ativa demonstra tratar-se de um profissional que busca constante aprendizado e sente-se feliz em compartilhar com o mundo seu conhecimento e descobertas. É claro que exceções existem, mas a regra é esta.

# CONCLUSÃO

Gerência de Configuração pode ser tratada sob diferentes perspectivas em função do papel exercido pelo participante do processo de desenvolvimento. Na perspectiva gerencial, a Gerência de Configuração é dividida em cinco funções, que são: identificação da configuração, controle da configuração, contabilização da situação da configuração, auditoria da configuração e gerenciamento de liberação e entrega. Sob a perspectiva de desenvolvimento, a Gerência de Configuração é dividida em três sistemas principais: controle de modificações, controle de versões e gerenciamento de construção. As cinco funções podem ser implementadas pelos três sistemas descritos na perspectiva de desenvolvimento, acrescidos de procedimentos manuais quando necessário.

Nesta seção, descreve-se os resultados obtidos a partir das implementações e estudos realizados com controle de versionamento utilizando github. Os resultados foram bastante satisfatórios levando o grau de complexidade baixo, porém em projetos maiores não mudaria muito, o principal já foi feito.

Neste trabalho relato a experiência de trabalhar com a prática de revisão de código. Assim, o Github permite publicar e ainda gerenciar de forma fácil todos os projetos, podendo criar um ou mais repositórios colocando diferentes projetos, não somente algoritmo como documentos entre outros.

# REFERÊNCIAS BIBLIOGRÁFICAS

PRESSMAN, Roger. **Engenharia de Software.** São Paulo: Pearson Makroh Books, 1995. 916 p.

SOMMERVILLE, Ian. **Engenharia de Software.** São Paulo: Pearson Makroh Books, 12007. 455 p.

FERREIRA, Gladstone. **Gerência de Configuração.** 2004. Disponível em: < http://www.cin.ufpe.br/~gfn/qualidade/gc.html>. Acesso em: 25 MAR. 2016.

BLOGGER, Rodger. [Contribuindo para projetos open-source no GitHub](http://rodger.com.br/blog/2013/08/17/contribuindo-para-projetos-open-source-no-github/). 2013. Disponível em: < http://rodger.com.br/blog/2013/08/17/contribuindo-para-projetos-open-source-no-github/>. Acesso em: 03 ABR. 2016.

SCHMITZ, Daniel. [Tudo](http://rodger.com.br/blog/2013/08/17/contribuindo-para-projetos-open-source-no-github/) que queria saber sobre o GitHub. 2015. Disponível em: < http://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar//>. Acesso em: 02 ABR. 2016.

# APÊNDICE A – Título do apêndice

Apêndices são documentos elaborados pelo(a) autor(a) deste trabalho.

# ANEXO A – Título do anexo

Anexos são documentos não elaborados pelo(a) autor(a) deste trabalho.