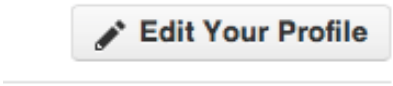
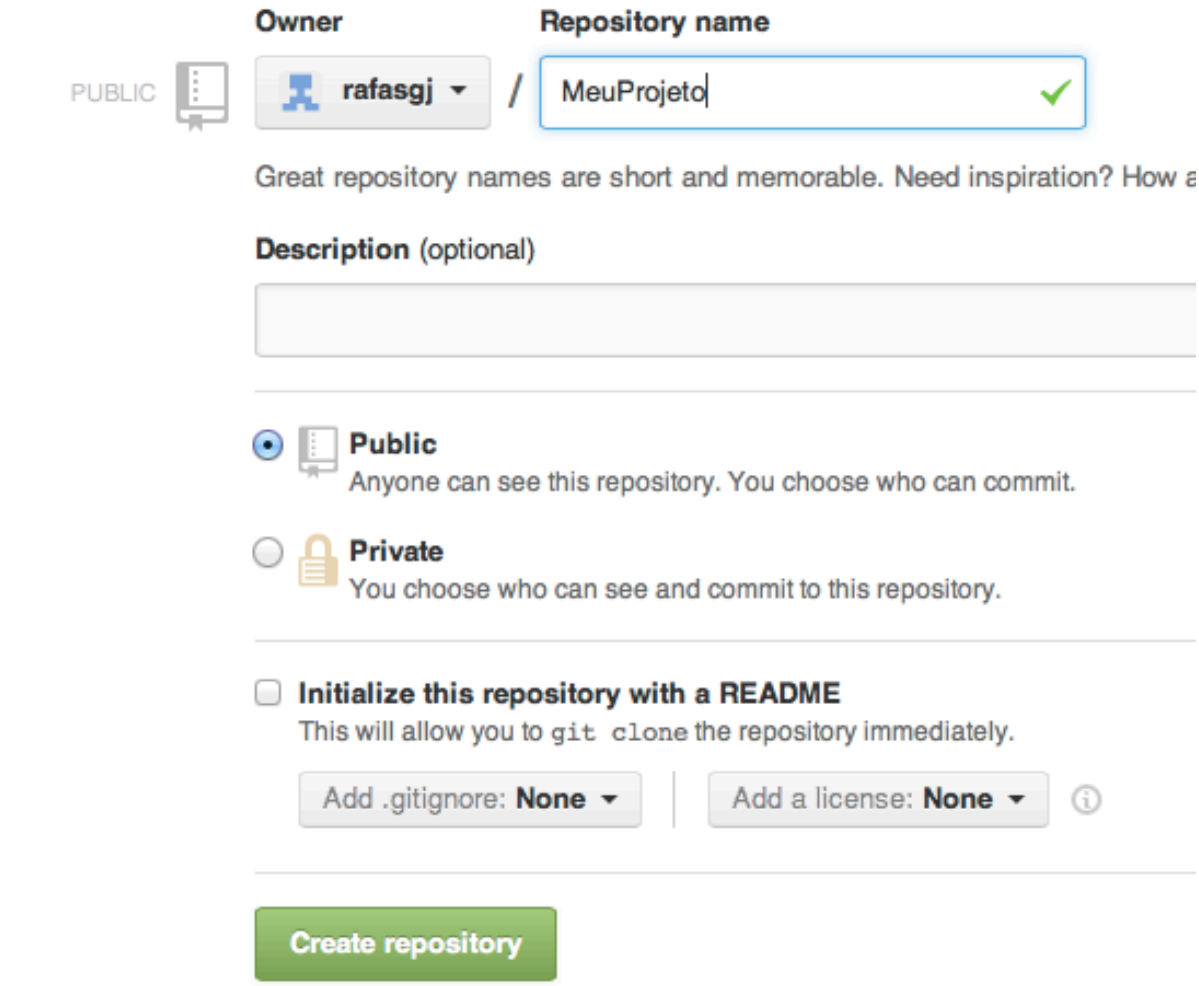
Uso do Git com o GitHub:

1) Como criar um repositório no GitHub?

Clique no botão "Create New Repo", para criar um novo repositório no GitHub.



Preencha os dados do repositório:



Obrigatoriamente, o repositório deve ter um nome, opcionalmente, uma descrição, e você tem a oportunidade de escolher se o repositórios será público, onde qualquer um pode acessar o código, ou privado, onde você terá a oportunidade de escolher com quem você quer compartilhar o repositório. Repositórios públicos são gratuitos, repositórios privados são pagos.

O Git não "gosta" de baixar repositórios vazios, logo, é interessante marcar a opção para inicializar o repositório com um "README", para facilitar o trabalho.

Opcionalmente, você pode escolher adicionar um arquivo ".gitignore" para a linguagem que você irá

utilizar. O ".gitignore" é um arquivo que lista os padrões de arquivos que o Git deve ignorar ao atualizar o repositório.

Uma explicação mais detalhada e alguns exemplos de arquivos ".gitignore" úteis podem ser encontrados em http://www.sujee.net/tech/articles/gitignore/.

Após configurar o seu projeto, clique em "Create Repository".

2) Como baixar um repositório?

Para baixar o repositório localmente para a sua máquina, você pode utilizar qualquer cliente Git. Embora existam diversos clientes gráficos para o Git, neste tutorial, serão abordados os comandos utilizados em um shell (prompt de camandos).

Antes de baixar o repositório, você deve saber o endereço do mesmo. Utilizaremos o protocolo HTTPS, por ser mais "tolerável" pro proxies e firewalls.

O endereço de um repositório do github é formado da seguinte forma:

    https://github.com/<USER>/<REPO>.git

Por exemplo, o repositório do projeto "Programa", do usuário "fulano", teria o seguinte endereço:

    https://github.com/fulano/Programa

Para baixar o repositório pela primeira vez, utilizamos o comando "clone" do Git:

    > git clone https://github.com/fulano/Programa

O problema do "clone" é que baixa todos os arquivos que um dia foram adicionados ao repositório, por esse motivo, não se deve adicionar arquivos temporários ao repositório.

Caso não seja possível baixar o repositório, pode ser necessário configurar o proxy que o git utiliza para buscar os arquivos. Para isso utilize o comando "config" com a chave "http.proxy", e passe o endereço IP do proxy e a porta:

    > git config --global http.proxy 192.168.0.1:8080

3) Como verificar os arquivos modificados?

É possível verificar quais arquivos foram modificados localmente com relação ao repositório local utilizando o comando "status".

    > git status

4) Como salvar as alterações localmente?

Para salvar as alterações dos arquivos no repositório local, utilize o comando "commit". Ao gravar as alterações, SEMPRE, insira uma mensagem sobre o que está sendo gravado, utilizando a opção "-m":

    > git commit -m "Correcao do bug #217."

5) Como atualizar o repositório remoto?

Para atualizar o repositório remoto, deve ser utilizado o comando "push", que guarda as modificações dos arquivos no repositório remoto:

    > git push <NAME> <BRANCH>

Onde "NAME" é o nome do repositório remoto, normalmente origin, que pode ser verificado com o comando:

> git remote

Ou criado com o comando:

> git remote add <NAME> <URL>

E "BRANCH", que é o branch de trabalho atual, normalmente "master", mas que pode ser criado como o comando:

> git branch <BRANCH>

6) Como atualizar a cópia local?

A atualização da cópia local dos arquivos a partir de um repositório remoto é uma tarefa executada em dois passos. Primeiro, devem ser recuperadas as alterações dos arquivos, após, deve ser feita a junção dessas alterações nos arquivos locais.

A tarefa de atualização da cópia local pode ser feita de duas formas diferentes.

A forma mais simples é executar o comando "pull", que irá executar os dois passos da atualização.

> git pull

A forma mais flexível é executar dois comandos, primeiro o "fetch" e após esse comando o "merge":

> git fetch

> git merge <NAME>

Esta forma é mais flexível, pois o "fetch" não altera os arquivos locais, e é possível, entre outras operações, verificar as alterações que serão feitas antes que sejam aplicadas localmente com o commando "diff":

> git diff

7) Trabalhando com Tags.

Tags são úteis para gerar "marcas" no repositório e operarmos em cima dessas marcas, ao invés das "hashs" criadas pelo Git, ou através de datas.

Para criar uma "tag leve", basta utilizar o comando "tag" da seguinte forma:

> git tag <TAG>

Todos os arquivos a partir do diretório no qual o comando está sendo executado receberão a tag especificada.

Para criar uma "tag anotada", utiliza-se a opção "-a" do comando "tag", junto com a opção "-m", já que uma tag anotada exige uma mensagem.

> git tag -a -m "Release para testes" <TAG>

Para exportar as tags para o repositório remoto, é preciso executar o comando "push", com a opção "--tags":

> git push --tags