**UNIVERSIDADE NOVE DE JULHO**

**DIRETORIA DOS CURSOS DE INFORMÁTICA**

**Giovanna Gonçalves Figueiredo - RA 919121050**

**Ricardo Oliveira Rufino – RA 919101838**

**Rafael Miecyslau Estefan Pereira - RA 919105745**

**CONTROLE DE VERSÃO**

**SÃO PAULO**

**2019**

**SUMÁRIO**

[1. OBJETIVOS 3](#_Toc24959404)

[2. CONTROLE DE VERSÃO 4](#_Toc24959405)

[3. EXEMPLOS DE FERRMENTAS DE CONTROLE DE VERSÃO 6](#_Toc24959406)

[4. GIT 7](#_Toc24959407)

[5. GITHUB 10](#_Toc24959408)

[6. CONCLUSÃO 15](#_Toc24959409)

[7. REFERÊNCIAS 16](#_Toc24959410)

# OBJETIVOS

O objetivo desta pesquisa é apresentar ao leitor de forma simples, clara e objetiva como funciona um sistema de controle de versão, sua finalidade, e suas ferramentas disponíveis, também falar sobre o Git e Github que é bastante usado por desenvolvedores em geral na maioria das empresas.

# CONTROLE DE VERSÃO

O que é?

Um sistema de controle de versões é um software usado para gerenciamento de projetos, ele auxilia os desenvolvedores no conhecimento de todos os processos ocorridos a um projeto, desde sua criação até cada modificação, suas finalidades são diversas visto que cada software de versionamento apresenta diferentes opções que devem ser escolhidas conforme a necessidade dos desenvolvedores.

O uso de tais ferramentas são cruciais no desenvolvimento, especialmente em grandes projetos onde diversas pessoas que podem ou não estar no mesmo ambiente de trabalho estão simultaneamente realizando alterações, o versionamento é extremamente importante para a organização.

Uma das principais finalidades desse software é mapear o projeto desde seu início, da versão 0.1.0 (primeiros passos), 1.0.0 (seu lançamento) e todas as atualizações decorrentes, por exemplo uma simples correção de bugs 1.01.0 ou alguma alteração importante 1.2.0, além disso os sistemas de versionamento sempre guardam uma cópia da versão anterior para que caso algo ocorra fora do normal, temos a segurança de ter uma versão estável guardada até que o erro seja corrigido.

Como funciona?

Cada sistema de controle de versão é diferente, porém, basicamente funciona assim:

As informações com o histórico ficam guardadas em um repositório em um servidor que pode ser acessado pela rede ou diretamente pela máquina do servidor, o repositório guarda a informação e todos os clientes podem consultar e escrever nesses arquivos, cada desenvolvedor possui uma cópia da ultima versão dos documentos e cada alteração deve ser atualizada no servidor submetendo (commit) as alterações, com isso os outros desenvolvedores devem sempre atualizar (update) o servidor para saber as novidades.

Sistemas de controle de versão Locais, Centralizados e Distribuídos.

O sistema de controle de versão local foi o primeiro a ser desenvolvido, pois, antigamente tudo era feito manualmente, as informações eram colocadas em um diretório de uma máquina local, porém, caso houvesse alguma falha de atenção esse diretório poderia ser perdido ou substituído, com o sistema de controle de versão local as alterações são controladas de forma local sob revisão.

O sistema de controle de versão centralizado foi criado visando principalmente o trabalho em grupo, esse sistema funciona na arquitetura cliente-servidor, o servidor armazena todas as informações e os clientes podem acessá-las, modifica-las e regata-las, esse sistema foi usado por muitos anos chegando a se tornar padrão, porém, o mesmo possui um risco, pois, como todas as informações estão centralizadas no servidor caso ocorra um acidente existe a possibilidade de toda documentação ser perdida.

Visando os problemas com os sistemas de controle de versão local e centralizado surgiu o sistema de controle de versão distribuído, que armazena os arquivos de forma local e no servidor, ou seja, se ocorrer qualquer problema no servidor todos os desenvolvedores terão todo o histórico salvo em sua máquina local, isso também possibilita que todos trabalhem simultaneamente no mesmo projeto, esse tipo de sistema é o mais usado atualmente.

# EXEMPLOS DE FERRMENTAS DE CONTROLE DE VERSÃO

Bitbucket

É uma ferramenta de controle de versão parecido com o github, porém, o mesmo possui a opção paga e grátis, sendo a opção grátis aberta para que todos possam ver e a paga é exclusiva, uma das vantagens é que o mesmo pode ser integrado ao trello que é uma ferramenta de organização, e eles também oferecem tutoriais para que o cliente possa aprender a mexer em seu sistema.

Visual Studio Team Services

Essa ferramenta além do controle de versão, possui vários outros recursos como os tickets, integração de códigos, ferramentas de planejamento, etc. é gratuita para até 5 usuários o que facilita para quem está começando seu primeiro projeto, possui integração das ferramentas de desenvolvimento e o mesmo é um serviço em nuvem.

Subversion

Esse sistema é a evolução do CVS, é um repositório centralizado que armazena todo o histórico, porém é necessário criar uma cópia local, é um sistema open-source que funciona em rede.

Mercurial

É uma ferramenta que opera no modelo distribuído, porém é necessário ter conhecimento também no modelo centralizado, seus comandos são muito parecidos com o subversion, e apesar de simples é uma ferramenta que permite fazer operações complexas.

Bazaar

É um sistema descentralizado, tem o proposito de acompanhar a evolução do sistema dando o máximo de liberdade ao desenvolvedor, é um software livre, permite que você trabalhe sem necessariamente estar conectado do servidor, pode ser usado no modelo centralizado ou distribuído e suporta ramificações vinculadas.

# GIT

O Git é um sistema open-source de controle de versão utilizado pela grande maioria dos desenvolvedores atualmente. Com ele podemos criar todo histórico de alterações no código do nosso projeto e facilmente voltar para qualquer ponto para saber como o código estava naquela determinada data.

Além disso, o Git nos ajuda muito a controlar o fluxo de novas funcionalidades, entre vários desenvolvedores no mesmo projeto com ferramentas para análise e resolução de conflitos quando o mesmo arquivo é editado por mais de uma pessoa em funcionalidades diferentes.

Tudo no Git é movido através dos pontos na história do projeto que são chamados de commits, esses pontos são formados por conjuntos de alterações em um ou mais arquivos e somados a um descritivo que resume as alterações nesse ponto.

De forma prática, pensando que tenhamos que desenvolver um sistema de login completo, nossos commits podem ficar dessa forma:

1. Configuração da estrutura do projeto

2. Estrutura da página de login

3. Estilos CSS da página de login

4. Estrutura da página de cadastro

5. Resolvido problema no login

6. Estilos CSS da página de cadastro

Veja que nossos commits descrevem exatamente as alterações que o código sofreu e além do título podemos detalhar ainda mais com um texto maior.

É muito importante essas informações estarem bem completas para que todos do time possam entender o que foi feito nesse ponto

Ramificações

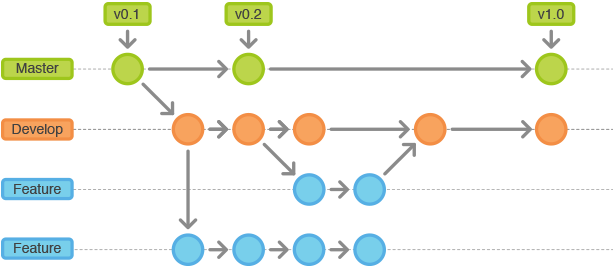
Imagine que você esteja trabalhando no meio de uma grande funcionalidade, pode levar até 2 meses para terminá-la. Em uma bela manhã de sol seu chefe resolve pedir urgentemente uma alteração na versão em produção da aplicação, ou seja, você não pode utilizar o código em que está trabalhando pois o mesmo possui features inacabadas. Como resolver?

As ramificações ou branchs no Git são formas de termos uma mesma versão do código sofrendo alterações e recebendo commits de diferentes fontes e inclusive por diferentes desenvolvedores.

Dessa forma, nós podemos manter um ramo para nossa funcionalidade que irá levar mais tempo e trabalhar em outro branch com a versão em produção para realizar alterações mais urgentes. E fica tranquilo, no fim de tudo o Git ainda vai nos ajudar a unir os códigos desses dois ramos de forma muito simpática.

Por padrão, você sempre está trabalhando em um ramo no Git, e mesmo quando você não cria um branch, o Git cria automaticamente um branch chamado master como padrão.

Na imagem abaixo podemos ver um exemplo de trabalho com vários ramos e commits aplicados. Veja que em alguns pontos da história os ramos são unidos para que as alterações de um ramo sejam aplicadas a outro.



Nesse caso, “master”, “Hotfix”, “Release”, “Develop” e os “Feature” são os ramos enquanto que os círculos são os commits. As caixas com v0.1, v0.2 e v1.0 são versões (conhecidas por tags) que foram pra versão em produção e podem ser compostas por pontos na história de vários branchs.

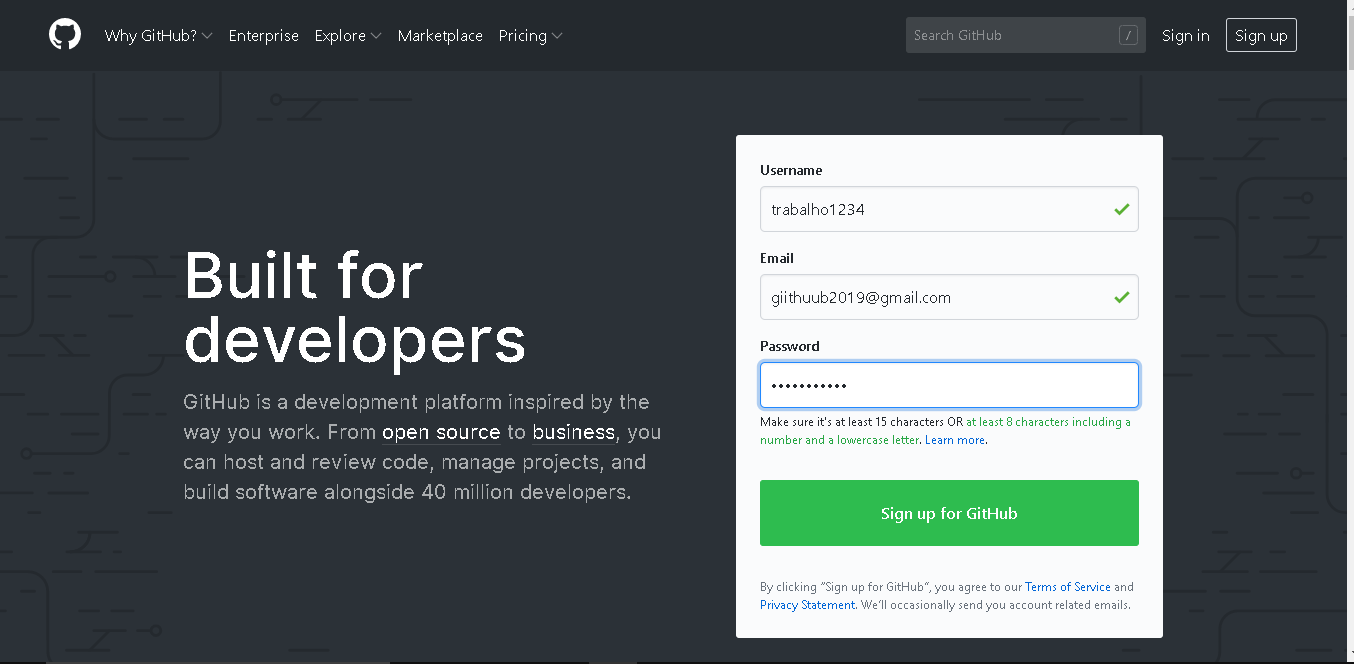
A diferença entre git e github é que o git é só uma ferramenta para versionar projetos, enquanto o github é o site no qual você colocará esses projetos versionados. uma analogia válida seria que o git é seu pincel e tintas enquanto o github é um museu.

# GITHUB

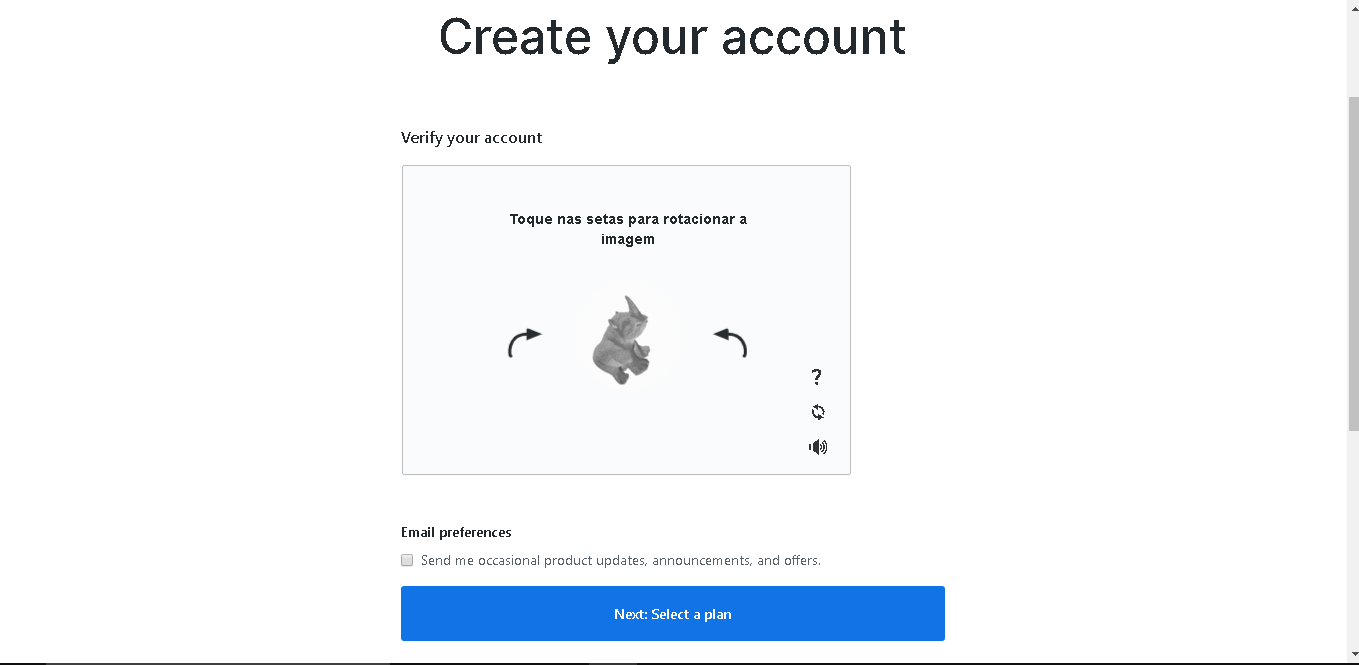
O GitHub é um serviço na web que possui diversas funcionalidades ligadas ao serviço Git. Você poderá usar free o GitHub para hospedar seus projetos pessoais porém se quiser deixar seus projetos privados (com acesso restrito) terá que pagar uma mensalidade. Além disso, quase todos os projetos/frameworks/bibliotecas sobre desenvolvimento open source estão no github e você pode acompanhá-los através de novas versões, contribuir informando bugs e também enviando códigos e correções. Se você é desenvolvedor e ainda não tem github, recomendo seriamente a ter um, para o seu próprio bem.

Criando uma conta

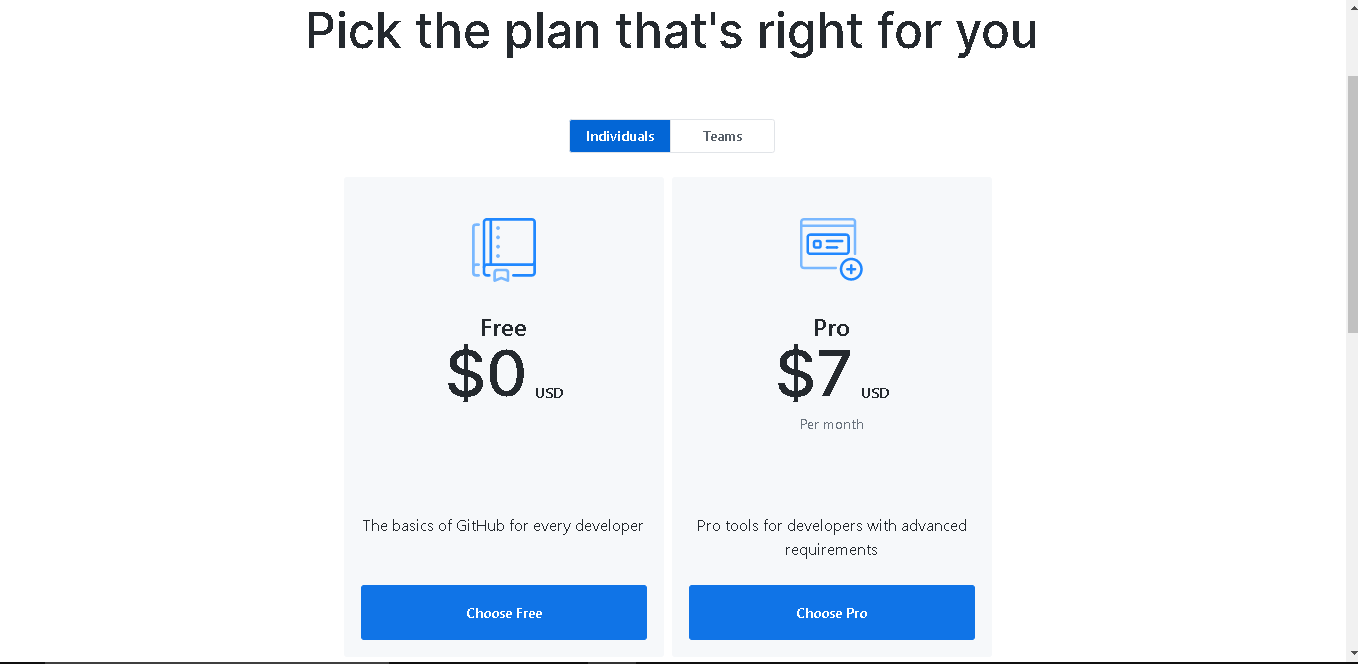
Para criar uma conta é bem simples, apenas precisa criar um nome de usuário, ter um e-mail e criar uma senha, como mostra a imagem abaixo:



O GitHub possui essa verificação de segurança apenas para ver se você é mesmo você e não um robô tentando destruir a humanidade. Segue imagem:

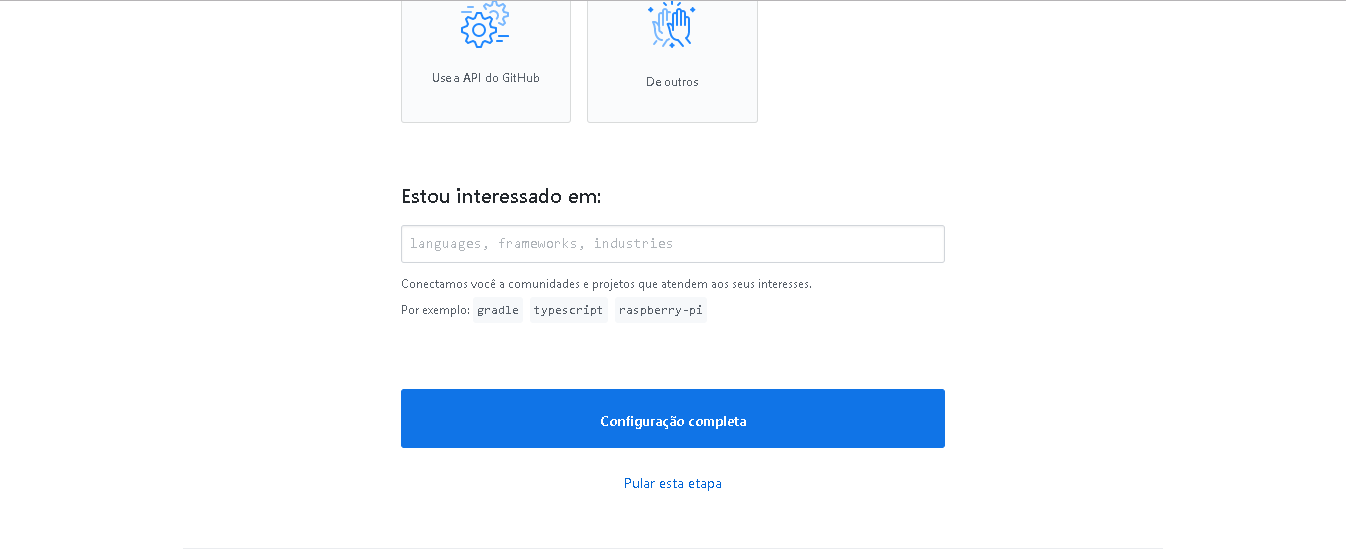
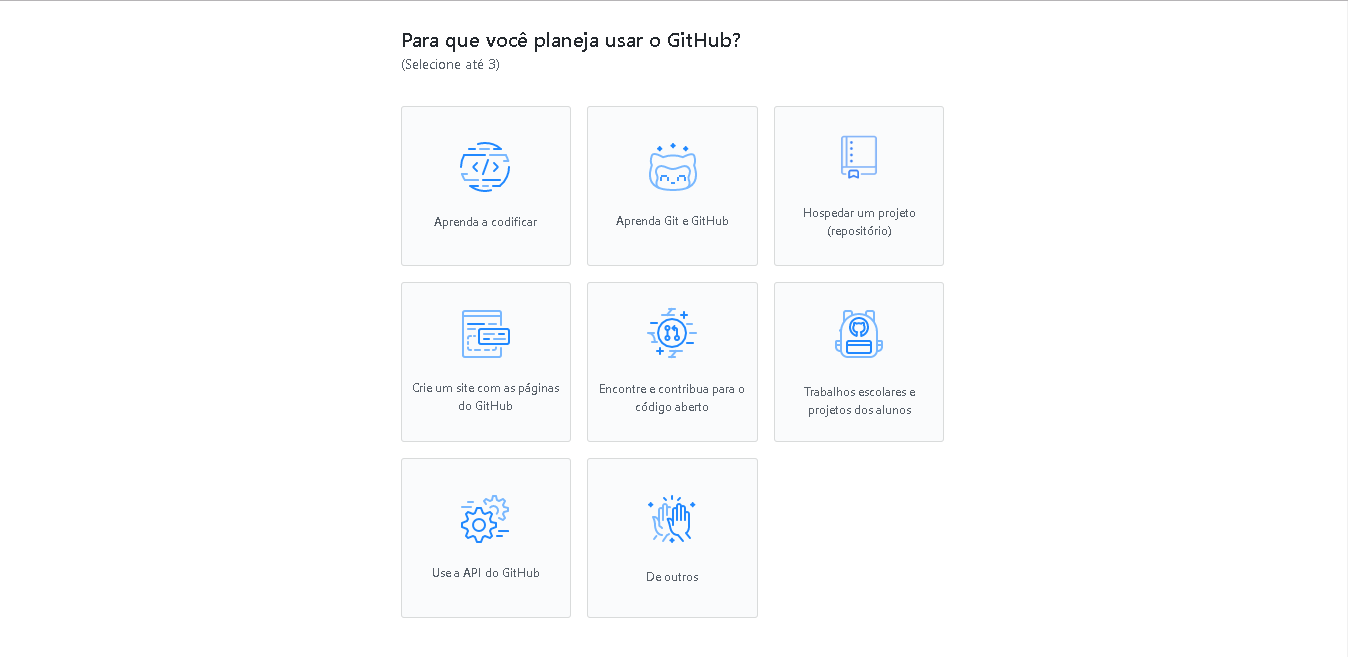


Após fazer a verificação de segurança é preciso escolher o seu plano que são compostos por “Free” e “Pro”, que como havíamos falado anteriormente o “Pro” é para manter seus projetos/códigos privados, segue imagem:

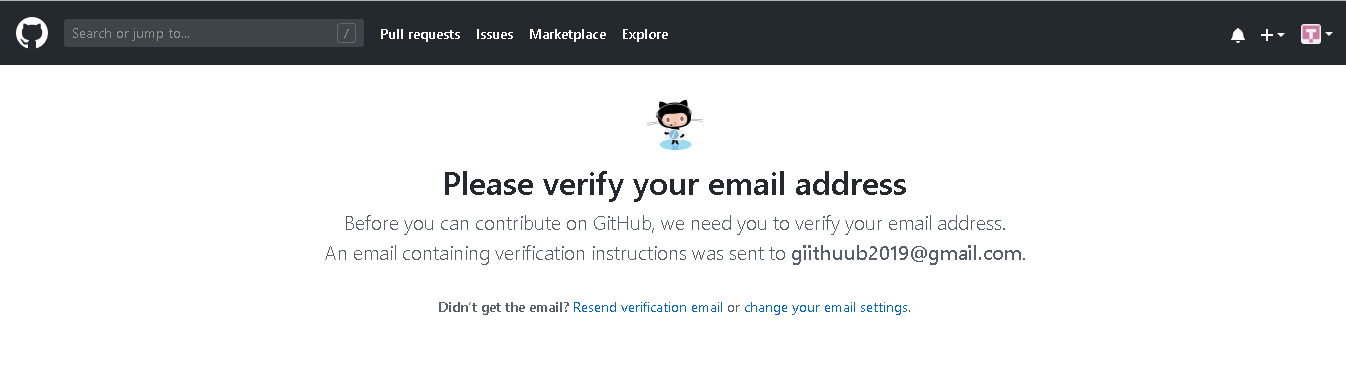


Em seguida é preciso responder a um mini questionário para o GitHub saber o que você precisa. Segue imagem:

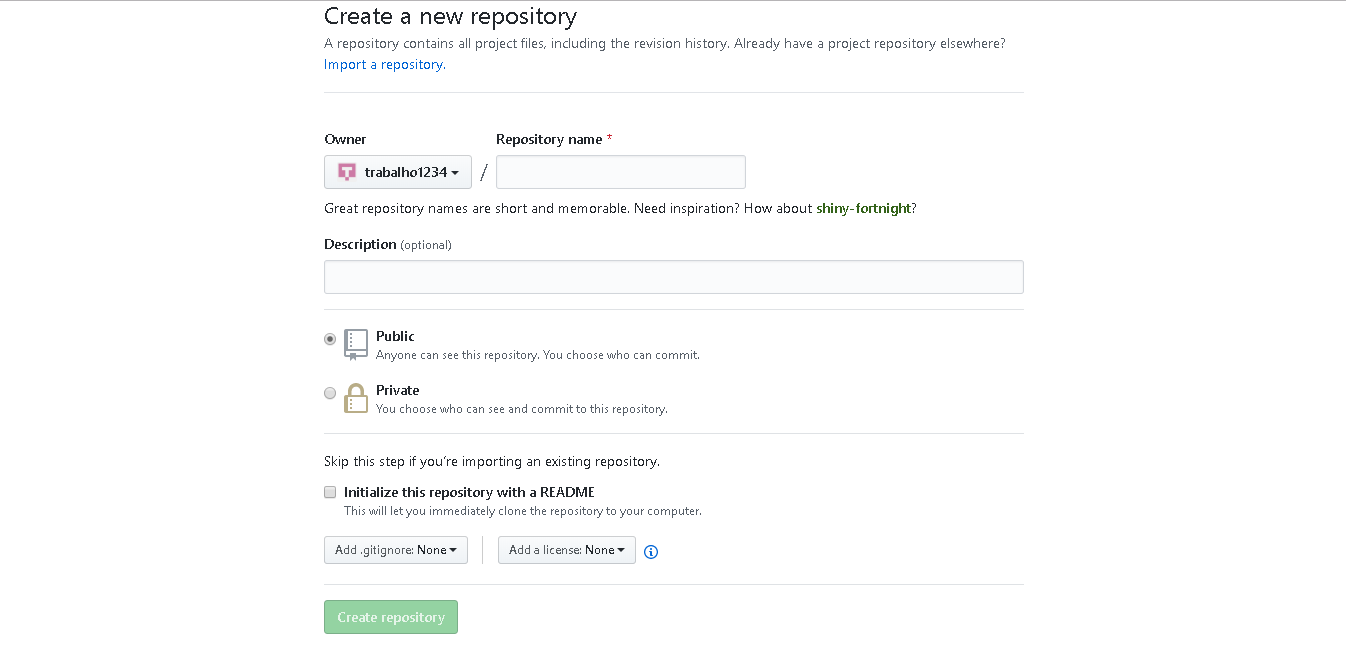




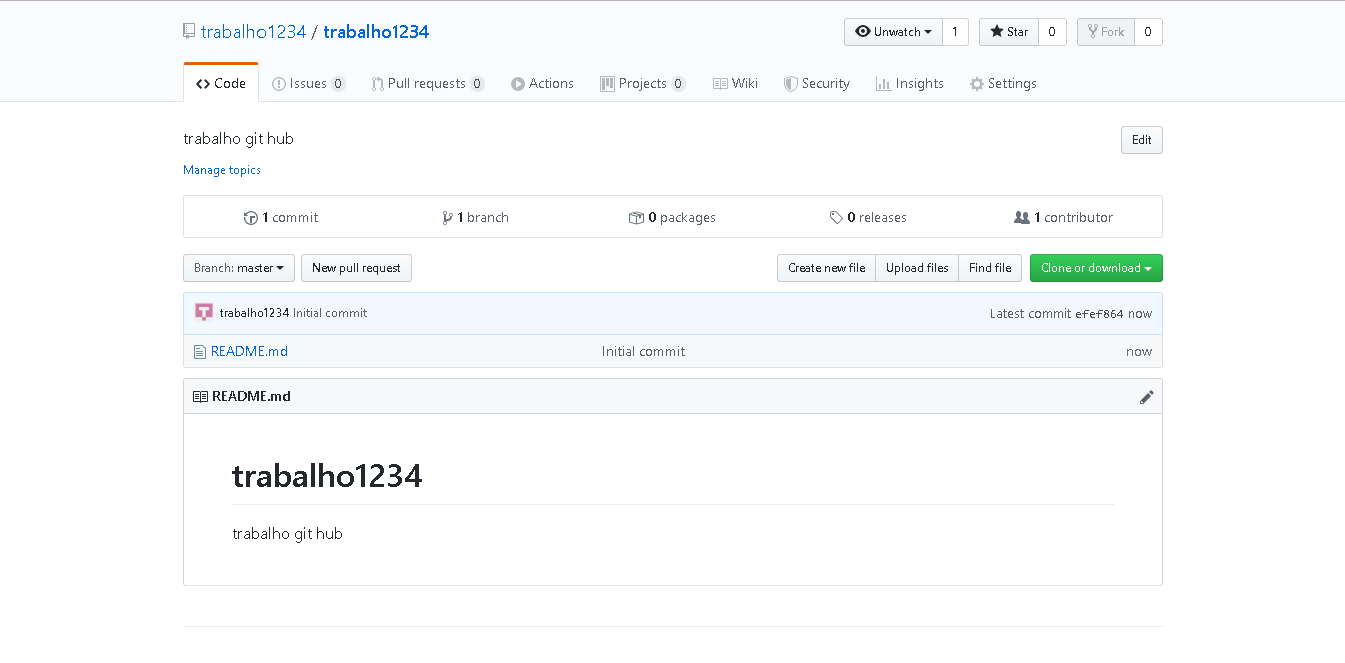
Feito isso o GitHub vai mandar um e-mail de verificação, basta abrir o e-mail e clicar no link contido nele que sua conta será ativada. Segue imagem:



Após confirmar seu e-mail, você precisará criar o seu Repository, esse onde o seu trabalho será armazenado. Você dará um nome a ele, uma descrição sobre o que se trata (caso queira) e selecionará se ele é Public ou Private (caso tenha o plano Pro). Segue imagem:



Ao terminar essas etapas, você está oficialmente apto a usar a plataforma do GitHub, meus parabéns e divirta-se! Segue imagem:



# CONCLUSÃO

Concluimos que o controle de versão é extremamente importante para o desenvolvimento de qualquer projeto, seja ele um simples projeto desenvolvido por apenas uma pessoa ou um grande projeto feito por uma equipe.

Suas vantagens são diversas sendo possível encontrar a ferramenta essencial para seu projeto de acordo com sua necessidade, o versionamento é indispensável para qualquer projeto.

# REFERÊNCIAS

<https://pt.wikipedia.org/wiki/Sistema_de_controle_de_vers%C3%B5es>

<https://medium.com/@johnsonmauro/sistemas-de-versionamento-de-softwarte-af132dfe7ff9>

<https://gaea.com.br/sabe-fazer-versionamento-de-software/>

<https://engineering.contaazul.com/versionamento-de-software-na-era-%C3%A1gil-8b53f6c08192>

<https://gaea.com.br/entenda-por-que-versionamento-de-software-e-tao-importante/>

<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o>

<https://digaotutoriais.wordpress.com/2016/04/12/controle-de-versao-localcentralizado-ou-distribuido/>

<https://www.youtube.com/watch?v=IW3t-NC2LAA>

<https://bitbucket.org/>

<http://www.ramonduraes.net/2015/12/06/introduo-do-visual-studio-team-services/>

<https://www.youtube.com/watch?v=j6sqmPQKkFQ>

<https://pythonhosted.org/sc.dev.core/vcs/subversion.html>

<https://pt.wikiversity.org/wiki/Subversion_-_SVN>

<https://blog.pronus.io/posts/qual-a-melhor-ferramenta-de-controle-de-versao-subversion-git-ou-mercurial/>

<https://blog.rocketseat.com.br/iniciando-com-git-github/>

<https://www.alura.com.br/artigos/comecando-com-git-aprendendo-versionar>

<https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/>