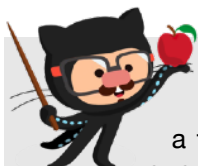


DevWeb

Capítulo 18

Aprendendo Git e GitHub

Talvez o conteúdo desse capítulo seja um dos que você mais vai usar durante a sua carreira, que está começando agora. Vamos aprender agora o que são os repositórios, tanto os locais quanto os remotos e qual é a diferença entre eles. E por fim, vamos aprender a colocar nossos projetos no ar, para que possamos acessar os sites em qualquer lugar com conexão à Internet.

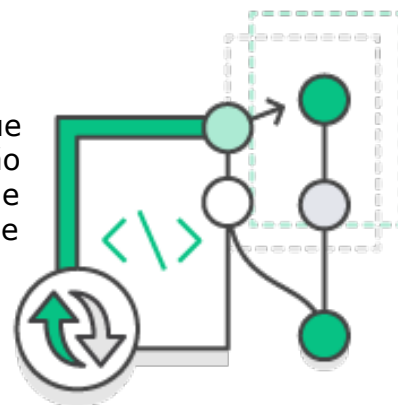


Você tem todo o direito de usar esse material para seu próprio aprendizado. Professores também podem ter acesso a todo o conteúdo e usá-los com seus alunos. Porém todos o que usarem esse material - seja para qual for a finalidade - deverão manter a referência ao material original, criado pelo **Prof. Gustavo Guanabara** e disponível no endereço do seu repositório público <https://github.com/gustavoguanabara/>. Este material não poderá ser utilizado em nenhuma hipótese para ser replicada - integral ou parcialmente - por autores/editoras para criar livros ou apostilas, com finalidade de obter ganho financeiro com ele.



Você precisa de um repositório para ser feliz

Talvez nesse exato momento você não faça ideia do que seja um **repositório** e esteja achando que minha afirmação acima foi um pouco exagerada, mas saiba que vou me esforçar para que você consiga entender o uso desse recurso e adote na sua vida de hoje em diante. Repositórios não vão te ajudar apenas no desenvolvimento em **HTML** e **CSS**, eles também serão úteis na criação de qualquer tipo de código em qualquer linguagem de programação que você vá decidir aprender de hoje em diante.



Para explicar a utilidade de cada um dos dois tipos de repositórios que vamos usar (**locais** e **remotos**), criei três pequenas histórias que podem acontecer na sua vida.

Situação 1: como guardar várias versões do seu site?

Quando estamos criando um projeto, começamos aos poucos criando a sua primeira versão dentro de uma pasta local em nosso computador.



Quando atingir uma versão estável, para manter um “*backup* seguro”, sempre usamos ideias brilhantes como gerar uma versão compactada dessa pasta. Isso vai permitir continuar a desenvolver uma versão aprimorada sem perder a versão anterior. Se por acaso algo der errado (e acredite, geralmente dá) basta descompactar nosso arquivo de versão estável, substituir o conteúdo da pasta original e recomeçar os trabalhos da criação da próxima versão. Usando essa prática, com certeza vai acabar se deparando com vários arquivos ZIP com nomes esquisitos.



Esses arquivos compactados ficam se acumulando no seu computador e são úteis caso você precise “voltar no tempo” e desfazer as bobagens que fizemos na madrugada passada quando tentamos desenvolver algo com muito sono.

Situação 2: seu HD foi pro saco, e agora?

Quem nunca perdeu arquivos de trabalho, que atire o primeiro HD queimado! Os dispositivos de hardware sempre param de funcionar no momento em que mais precisamos. E se utilizamos apenas a técnica descrita na situação anteriormente, é

possível que todo o trabalho seja perdido em um estalar de dedos.

Uma das saídas utilizadas é salvar cópias de segurança em *pendrives* e em HDs externos, além de manter um *backup* em serviços de armazenamento como **Dropbox**, **Google Drive**, **OneDrive**, **iCloud**, etc.



O problema de manter esses *backups* isolados é que, no momento de ligar nossa “máquina do tempo” e voltar para uma versão anterior, precisamos fazer o *download* do ZIP desejado e substituir manualmente a versão atual para a última estável.

Situação 3: mostrar seu site para o mundo

E o que acontece quando você precisar mostrar o projeto de um site que você criou para um amigo, para seu professor ou até mesmo para um possível cliente? Vai mandar o link de um arquivo ZIP para ele(a) baixar, descompactar e abrir no navegador local? É assim que as pessoas estão acostumadas a acessar sites?

Os serviços de armazenamento que eu citei na situação anterior são muito úteis para fazer *backup*, mas possuem uma grande limitação quando queremos praticidade. Por exemplo, salvar os arquivos de um site no **Dropbox** não vai garantir de forma alguma que ele vai ficar disponível para o acesso externo. Ninguém vai poder “acessar seu site”, apenas digitando uma URL no navegador. Tudo vai ficar armazenado como um grupo de arquivos, não como um site hospedado.



ATÉ DÁ, MAS... Existe um recurso do **Google Drive** para a hospedagem de sites, mas nada muito simples e intuitivo. A melhor maneira é sempre usar serviços especializados em hospedagem de sites.

Os repositórios vão resolver isso

Todas as três situações descritas podem ser resolvidas com repositórios locais/remotos de maneira muito simples. E se por acaso você já tentou aprender sobre isso, achou guias com dezenas de comandos e não conseguiu se adaptar, saiba que tudo evoluiu muito nos últimos meses e vamos ver tudo de uma maneira muito simples, sem decorar nenhum comando! Para começar, vamos aprender que existem dois tipos de repositórios: os **locais** e os **remotos**.

Repositório Local

Um **repositório local** tem esse nome porque vai estar sempre no seu computador. A função dele é facilitar a gestão de diversas versões do seu projeto de forma simples e automática.



Lembra da **situação 1** que descrevemos anteriormente? Pois quando instalamos um **Sistema de Controle de Versões** (do inglês *Version Control System* - **VCS**), deixamos a responsabilidade por gerenciar as versões dos nossos projetos nas mãos de um *software* especializado, que vai fazer tudo automaticamente e vai permitir que você volte a qualquer ponto no momento em que achar melhor.

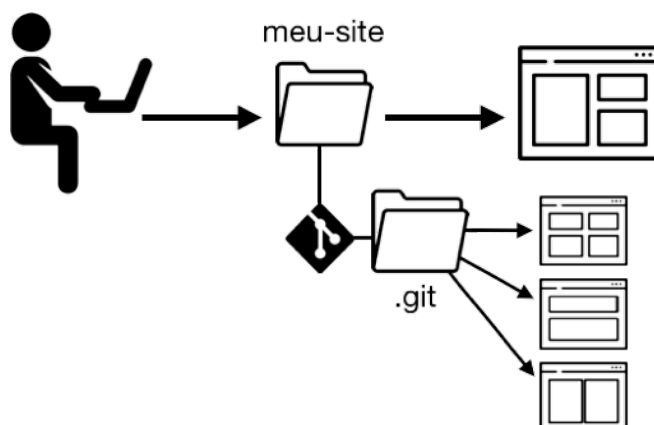
Um dos *softwares* VCS mais famosos é o **Git**, feito por **Linus Torvalds**, o mesmo criador do núcleo **Linux** dos sistemas operacionais. O sistema do Git foi criado no ano de 2005 em poucos dias (10 dias, para ser mais exato), por conta de uma briga entre o *Linus* e o criador de um software chamado **BitKeeper**, que era utilizado para gerenciar as versões em desenvolvimento do *Linux*.



QUER SABER MAIS SOBRE A TRETA? Não vou usar espaço desse material para contar a treta entre **Linus Torvalds** e **Larry McVoy**, mas já contei essa história no YouTube. Se você ficou curioso(a) para saber o que rolou, acesse o link a seguir:

Curso de Git e GitHub: <https://youtu.be/CJtrNuTTs4Q>

O esquema de funcionamento do Git é totalmente focado no nosso computador. Analisando a imagem a seguir, vemos que o software está monitorando uma pasta chamada *meu-site* que tem a versão atual do projeto e uma pasta especial chamada *.git* com várias versões pelas quais o site passou durante sua evolução.

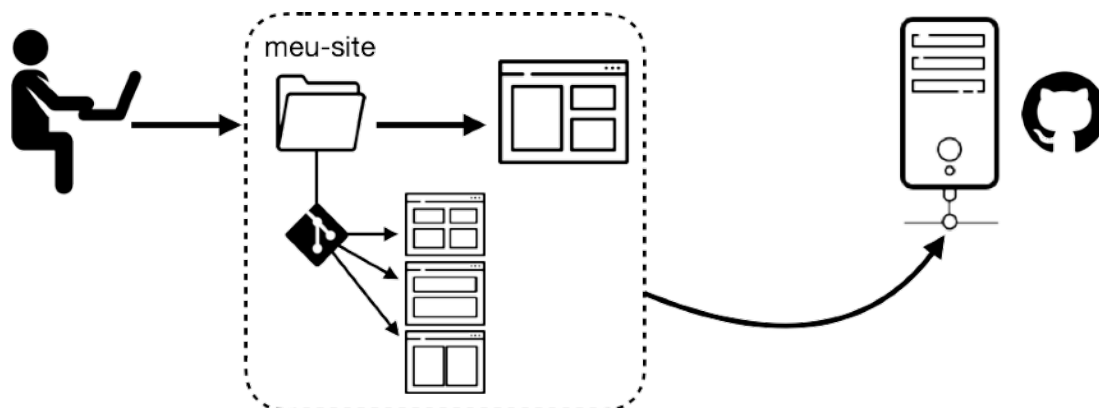


A grande vantagem de usar um **sistema de controle de versões** como o **Git** é poder voltar a qualquer momento para qualquer versão anterior do projeto de forma imediata, tudo 100% transparente para o programador.

Como você já deve ter percebido, os repositórios locais resolvem o problema que apontamos na **situação 1**, mas se o nosso PC quebrar (**situação 2**) ou se quisermos mostrar o projeto para outra pessoa (**situação 3**), ainda não é possível usando esse tipo de sistema, mas não se desespere agora! Continue lendo...

Repositório Remoto

Para solucionar os problemas levantados nas duas últimas situações desse capítulo, vamos precisar de um lugar na nuvem para guardar nossos repositórios locais. E é aí que entra o nosso segundo grande personagem: o **GitHub**.



Criado em 2008 por quatro amigos, o **GitHub** é um serviço que nos permite criar um **repositório remoto** na nuvem para guardar nossos projetos e versionamentos. Ele não é a única opção que existe no mercado (ainda temos o **GitLab**, o **Bitbucket**, e muitos outros) mas provavelmente é a mais popular, tanto que hoje pertence à gigante **Microsoft**, que comprou o serviço em 2018 por 7.5 bilhões de dólares.

Com o tempo, o GitHub começou a ganhar funcionalidades extras, que foram transformando o serviço em uma grande **rede social para programadores**. Além de guardarmos nossos códigos nos servidores, podemos nos comunicar com outros desenvolvedores e até mesmo colaborar com outros projetos que estão disponíveis publicamente para todos.



Ao colocar nossos códigos na nuvem, automaticamente resolvemos o problema da **situação 2**. E um recurso chamado **GitHub Pages** vai permitir a hospedagem gratuita de sites simples, que usem HTML + CSS + JS e disponibilize o acesso através de uma URL. Por exemplo, o mini-projeto que construímos no capítulo anterior está disponibilizado no endereço a seguir:

<https://professorguanabara.github.io/projeto-android/>

O código original também está disponibilizado publicamente no endereço abaixo:

<https://github.com/professorguanabara/projeto-android>

Como instalar e usar isso tudo?

Para poder aproveitar todos esses maravilhosos recursos que explicamos até aqui, precisamos instalar dois softwares no nosso computador: o **Git-SCM** e o **GitHub Desktop**. Também precisamos criar um perfil gratuito no site **GitHub**.

Para fazer o download e instalação do **Git**, acesse o site git-scm.com e clique sobre o botão de *download* da versão mais recente e instale o programa.

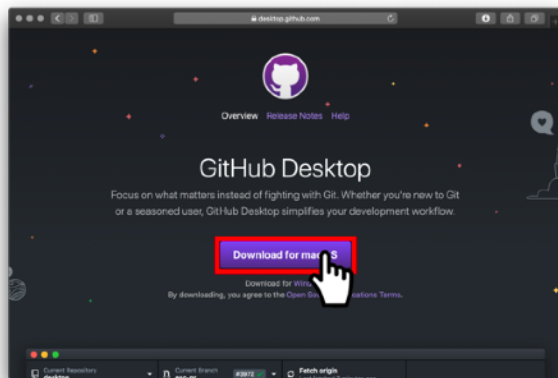
Nesse site, encontramos as versões para todos os sistemas: Windows, Linux e MacOS.

Para a instalação no Linux, o site mostra os comandos necessários para realizar a instalação via gerenciador de pacotes.



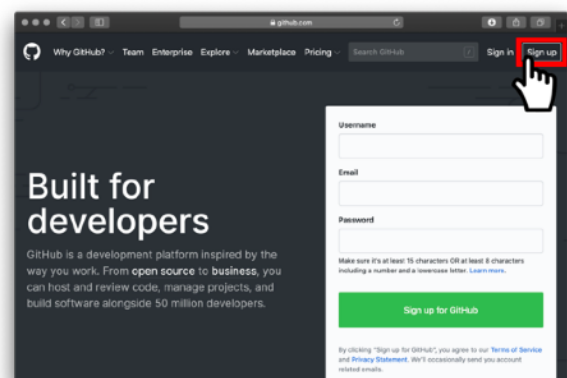
Para fazer o download e instalação do programa **GitHub Desktop**, acesse o site desktop.github.com e clique sobre o botão de *download* da versão atual e instale.

O GitHub Desktop também existe em versões para os sistemas Windows, Linux e MacOS. A versão para Windows só existe para sistemas 64-bits. Em Linux, devemos instalar via repositório.



Depois de instalar as duas ferramentas acima, acesse o site github.com e crie o seu perfil clicando no botão **Sign up** indicado na imagem a seguir. Você só vai precisar de um nome de usuário, e-mail de contato e uma senha segura. Um e-mail de verificação será enviado para ativar sua conta.

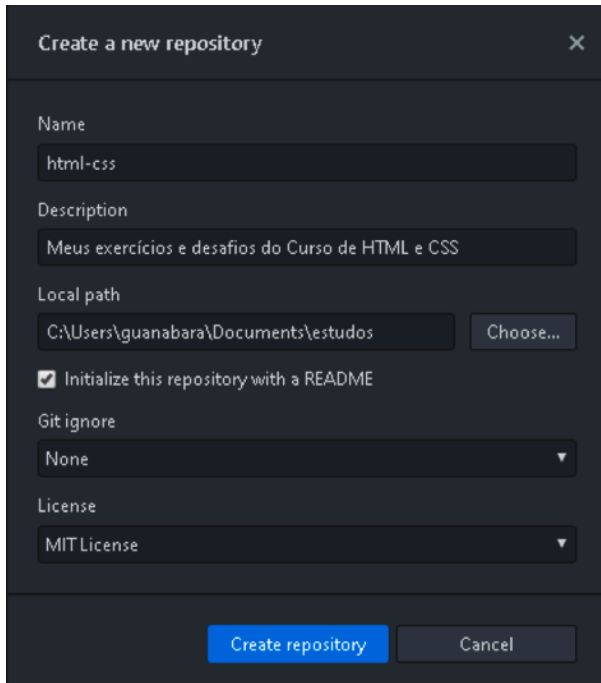
Se já tiver uma conta criada, clique sobre o botão **Sign in** ao lado.



FAÇA ESCOLHAS SÁBIAS! Tenha muito cuidado na hora de escolher seu nome de usuário. Ele fará parte do seu futuro profissional e pode ser que você precise divulgar seu perfil em uma entrevista de emprego. Fuja daqueles nomes engraçadinhos e agressivos.

Criando o seu primeiro repositório

Abra o **GitHub Desktop** (com conexão ativa à Internet, claro) e faça login na sua conta criada no **GitHub** diretamente na tela do programa. Depois de confirmar a conexão, crie um novo repositório em **File > New Repository** ou pressione **Ctrl+N**.



The screenshot shows the 'Create a new repository' window. The 'Name' field contains 'html-css'. The 'Description' field contains 'Meus exercícios e desafios do Curso de HTML e CSS'. The 'Local path' field shows 'C:\Users\guanabara\Documents\estudos' with a 'Choose...' button. The 'Initialize this repository with a README' checkbox is checked. The 'Git ignore' dropdown is set to 'None'. The 'License' dropdown is set to 'MIT License'. At the bottom are 'Create repository' and 'Cancel' buttons.

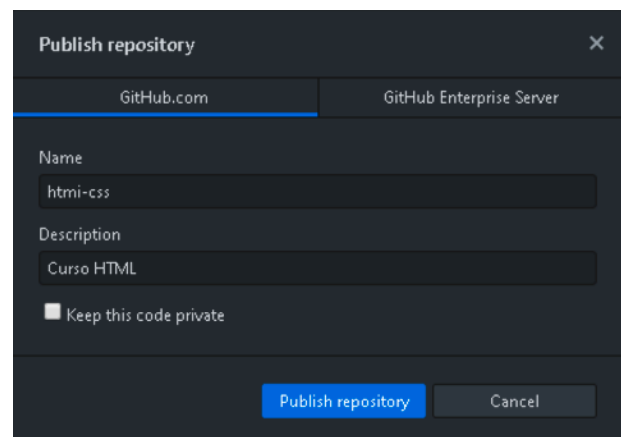
Para começar a versionar os arquivos que criamos durante o curso, vamos criar o repositório de nome **html-css** (mesmo nome da pasta que criamos) dentro da pasta **estudos** que está dentro dos nossos Documentos. Não se esqueça de marcar a opção para criar o arquivo **README.md** e escolher uma licença. No fim, clique em **Create repository**.

Este procedimento vai criar nosso **repositório local** via **Git**.

Depois de tudo, clique no botão **Publish Repository** para criar o nosso **repositório remoto** no **GitHub** e todos os arquivos serão enviados para a nuvem.

Assim que solicitar a publicação do repositório, deve escolher se vai deixá-lo **público** ou **privado** (padrão). Desmarque a opção *"Keep this Code private"*, principalmente se deseja utilizar o recurso do **GitHub Pages** e hospedar gratuitamente seu site HTML. Só projetos públicos podem ser hospedados e disponibilizados para outros.

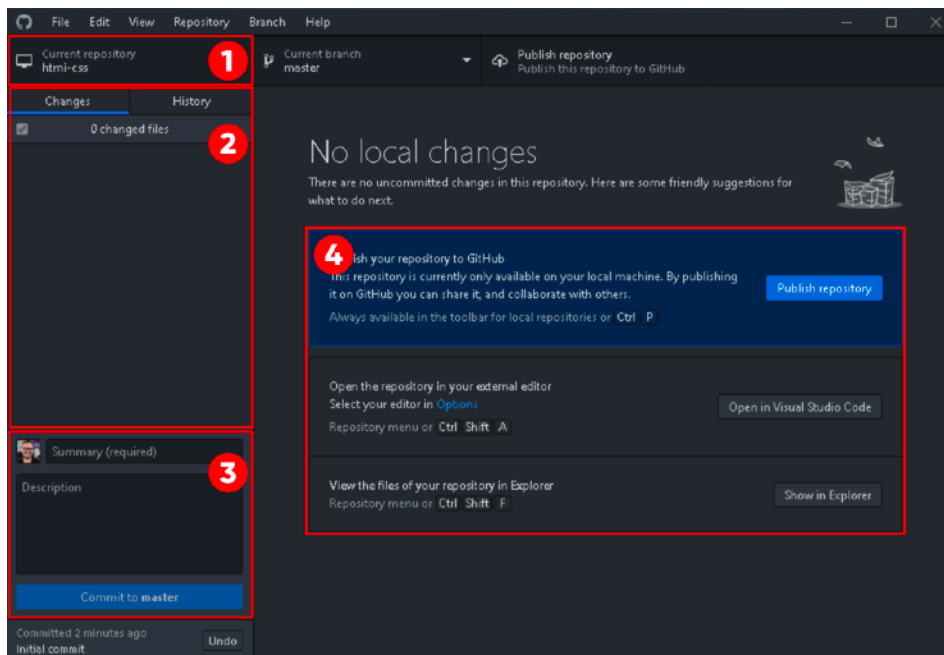
A interface do **GitHub Desktop** é bem simples, mas concentra muitas funcionalidades que unificam recursos do **Git** e do **GitHub** em um só lugar.



The screenshot shows the 'Publish repository' window. The 'GitHub.com' tab is selected. The 'Name' field contains 'html-css'. The 'Description' field contains 'Curso HTML'. The 'Keep this code private' checkbox is unchecked. At the bottom are 'Publish repository' and 'Cancel' buttons.



MUITO CONFUSO? TEREMOS VÍDEOS! Não se desespere se não estiver conseguindo realizar os procedimentos seguindo apenas esse material escrito. Ele será complementado com uma série de vídeos que estão sendo cuidadosamente criados pra você.



Na **área 1**, escolhemos o repositório ativo no momento. Clicando nessa área, podemos mudar entre todos os repositórios locais que temos no computador.

Já na **área 2**, vão aparecer todas as mudanças detectadas pelo Git nos arquivos que estão na pasta do repositório.

Usaremos a **área 3** sempre que quisermos estabelecer uma versão estável ou submeter uma alteração importante do nosso projeto. Ao colocar uma descrição na caixa **Summary** (resumo) e clicar no botão **Commit** (pode ser entendido como **compromisso** ou como **enviar**). Fazer um *commit* ou “*comitar*” (jargão técnico em Português) vai criar uma versão **local** do repositório usando o Git.

Depois de *comitar* um repositório local, podemos realizar uma operação de **Push** (empurrar) que vai transferir todos os códigos para o **GitHub**. O botão *push* vai aparecer na **área 4** logo após um *commit*. Também podemos realizar uma operação dessas clicando no menu Repository > Push ou ainda pressionar Ctrl + P.

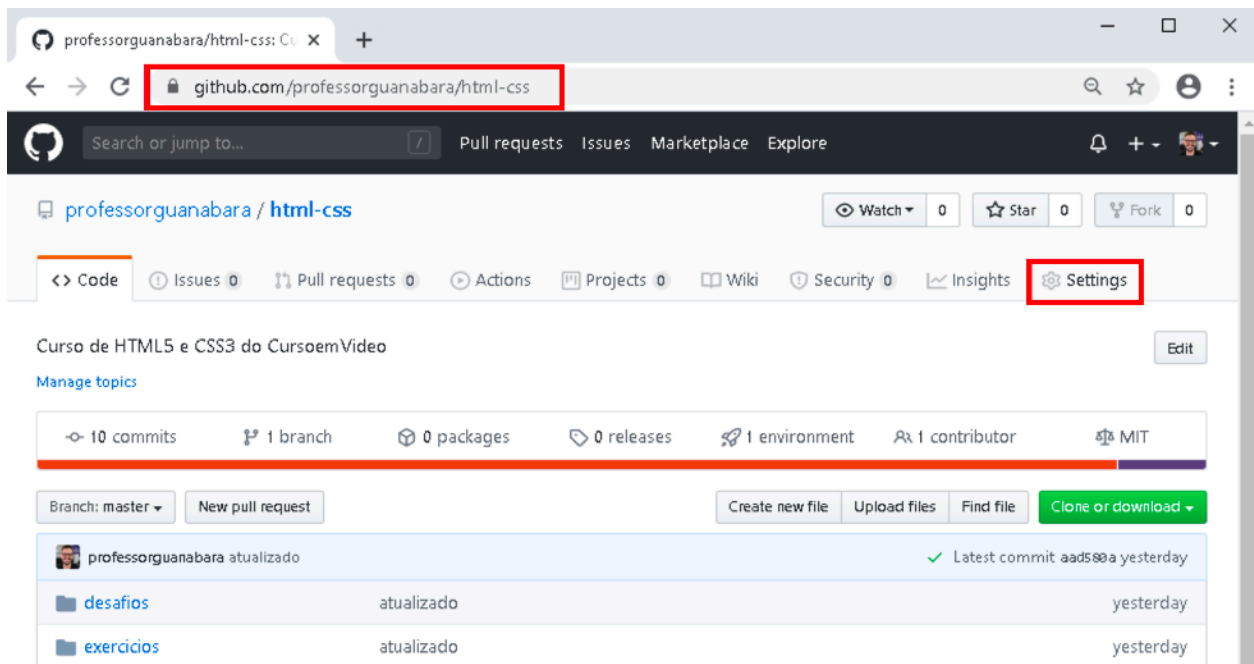
Na **área 4**, também vão aparecer botões muito importantes para abrir o projeto usando o Visual Studio Code ou abrir o gerenciador de arquivos do seu sistema com a pasta do projeto.

Habilitando a Hospedagem Grátis

O recurso do **GitHub Pages** não está ativo por padrão em nossos repositórios remotos, mas é muito fácil de habilitar. Isso só pode ser feito diretamente no site do GitHub, individualmente para cada repositório que queremos usar para hospedar projetos HTML.

Acesse o seu perfil no site <https://github.com/nomeusuario> (substituindo pelo nome do seu usuário, claro) e abra o repositório que quer habilitar a hospedagem. Em seguida, clique sobre a aba **Settings** e role a tela até a área **GitHub Pages**.

A imagem a seguir mostra a tela principal do repositório no site. Observe atentamente as áreas demarcadas, pois elas mostram a posição dos componentes.



Na área de opções, existem duas maneiras de habilitar o GitHub Pages. Na primeira, clique no botão **None** e escolha a opção **master branch**. Com essa maneira, você vai precisar ter um arquivo `index.html` com o código da página principal que será exibida ao acessar o site. Essa técnica é mais usada quando queremos hospedar um site único e receber uma URL para apresentar o projeto a outra pessoa.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

None ▾

Theme Chooser

Select a theme to publish your site with a Jekyll theme using the master branch. [Learn more.](#)

Choose a theme

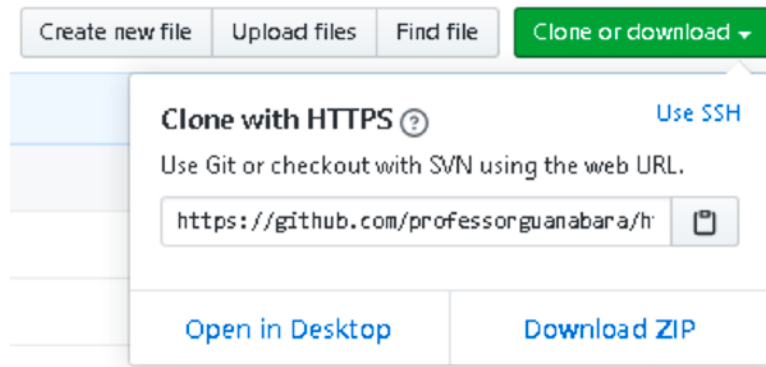
A segunda maneira é clicando sobre o botão **Choose a theme**, que vai permitir escolher um tema para exibir o conteúdo do arquivo `README.md` como uma página. Essa segunda técnica é mais usada quando vamos hospedar vários projetos em um mesmo repositório ou quando vamos guardar informações sobre um projeto que não é um site, como download de instaladores e coisas do tipo.



IMPORTANTE: O **GitHub** pode ser usado para guardar códigos em todas as linguagens, incluindo o PHP. Porém, o recurso **GitHub Pages** não vai servir para hospedar o projeto, gerando link para acesso à sua execução. Ele só serve para hospedar HTML + CSS + JavaScript.

Clonando um repositório

Outra operação essencial para quem vai usar o **GitHub** para hospedar repositórios remotos é **clonar** um projeto. Para isso, devemos acessar a URL do repositório no site github.com e procurar o botão **Clone or download** ao lado direito da tela.



Ao clicar sobre o botão verde, escolha a opção **Open in Desktop** e o programa **GitHub Desktop** vai abrir automaticamente e os arquivos serão baixados para o seu computador, criando um repositório local.

Podemos clonar qualquer projeto do **nosso** repositório remoto (público ou privado) ou até mesmo repositórios públicos de **outras pessoas**. Se o repositório for seu, você ainda vai poder atualizá-lo com as alterações que fizer. Já para repositórios de outras pessoas, você não vai poder efetuar alterações e efetuar operações de *push*, a não ser que o proprietário autorize (através das *Pull Requests*).

Eu já falei sobre isso no YouTube?

Eu sei que às vezes as pessoas gostam mais de assistir vídeos do que ler livros, e é por isso que eu lanço há anos materiais no canal Curso em Vídeo no YouTube. O link que vou compartilhar contigo tem o conteúdo explicado como você leu aqui, só que de forma mais ilustrada. Reserve um tempo dos seus estudos para assistir esse vídeo todo.



Curso em Vídeo: https://www.youtube.com/playlist?list=PLHz_AreHm4dm7ZULPAmadvNhH6vk9oNZA