

MATERIAL DE APOIO

GIT

Git é um **software de versionamento** de projetos muito utilizado na programação. Com ele podemos criar um histórico de alterações dos arquivos e estabelecer uma linha do tempo, em que o ponto mais recente é a **versão** mais atual do projeto.

A grande vantagem de um projeto estar versionado é ter a chance de retornar para uma versão anterior, caso a atual apresente problemas. Assim podemos **corrigir códigos** que causaram bugs e estabelecer uma nova versão de forma segura.

Além disso, a linha do tempo pode ser extremamente útil para que uma série de modificações no projeto fique **documentada**. Deste modo o entendimento das mudanças no decorrer do tempo é mais simples para quem está conhecendo ou vai iniciar em um projeto que já está em andamento.



Para projetos em equipe, o Git fornece outra grande vantagem. Com ele conseguimos criar partes diferentes de um projeto ao mesmo tempo, em conjunto com outros desenvolvedores. A junção das partes e a resolução de eventuais conflitos é facilitada pelo Git.



O Git cria no seu computador o projeto com os arquivos que você quer salvar, o qual chamamos de **repositório local**. Neste repositório, nada é compartilhado na internet. Ou seja, todas as alterações feitas no repositório irão permanecer no próprio computador. Para facilitar o compartilhamento das alterações entre mais usuários, é necessário usar uma plataforma online de versionamento a fim de criar um repositório remoto.

FAZENDO DOWNLOAD DO GIT

Nesta seção será apresentado o passo-a-passo de **instalação do Git para o ambiente Windows**. Caso seu sistema operacional seja outro, acesse os links abaixo para maiores informações:

- [Git Guides - install git · GitHub](#)
- [Instalando o Git](#)

INSTALANDO O GIT

1 - Acesse o site <https://git-scm.com/> e clique em “Download for Windows”, do lado direito da tela.





2 - Selecione a opção “**Click here to download**”, com a versão do Windows 64-bit. Caso o seu Windows seja 32-bit, selecione a opção mais abaixo com 32-bit:

git --fast-version-control

Search entire site...

WINDOWS PADRÃO 64-BIT

About Documentation Downloads Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Download for Windows

Click here to download the latest (2.35.0) 64-bit version of Git for Windows. This is the most recent [maintained build](#). It was released 3 days ago, on 2022-01-24.

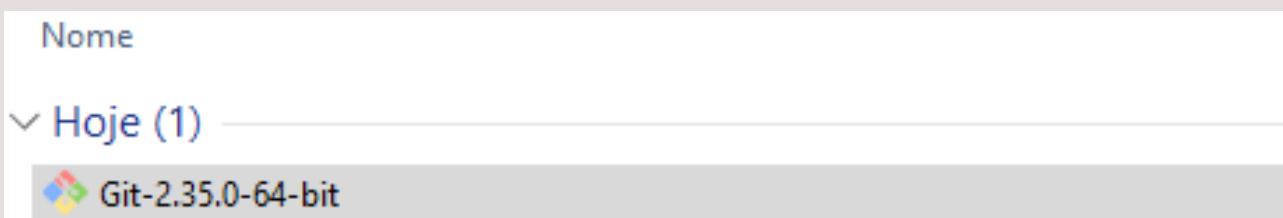
Other Git for Windows downloads

Standalone Installer
32-bit Git for Windows Setup.
64-bit Git for Windows Setup.

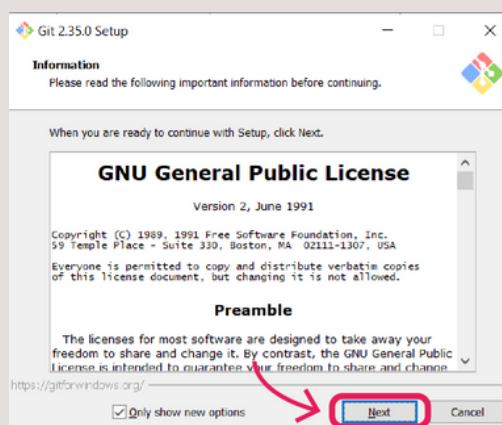
Portable ("thumbdrive edition")
[32-bit Git for Windows Portable](#).
[64-bit Git for Windows Portable](#).

SE SEU WINDOWS FOR FOR 32-BIT

3 - Execute o arquivo baixado para abrir a instalação, como no exemplo abaixo:



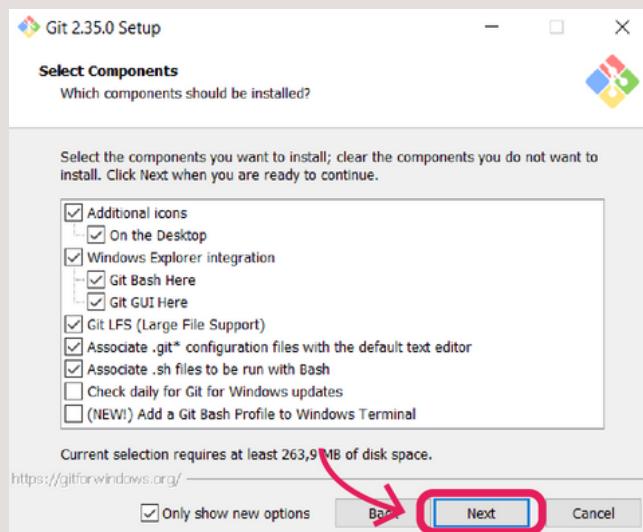
4 - Ao executar, a tela de instalação irá abrir. Pode ser preciso autorizar que o Windows execute a instalação; se precisar, confirme. Ao aparecer a primeira tela, clique em “**Next**”:



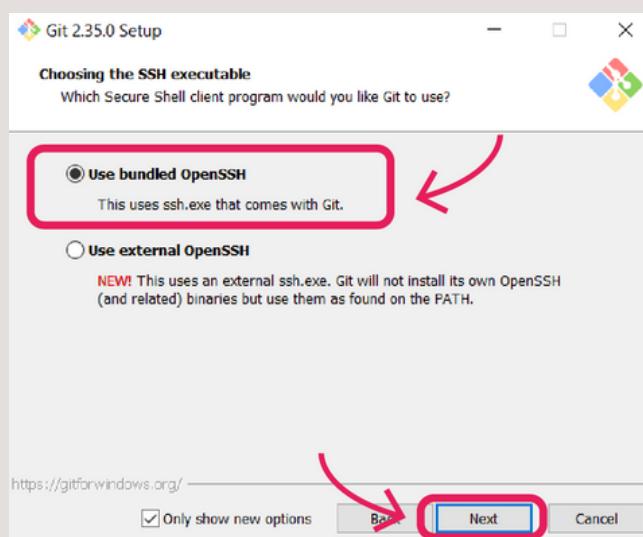


Versões de 32 bits e 64 bits do Windows: perguntas frequentes

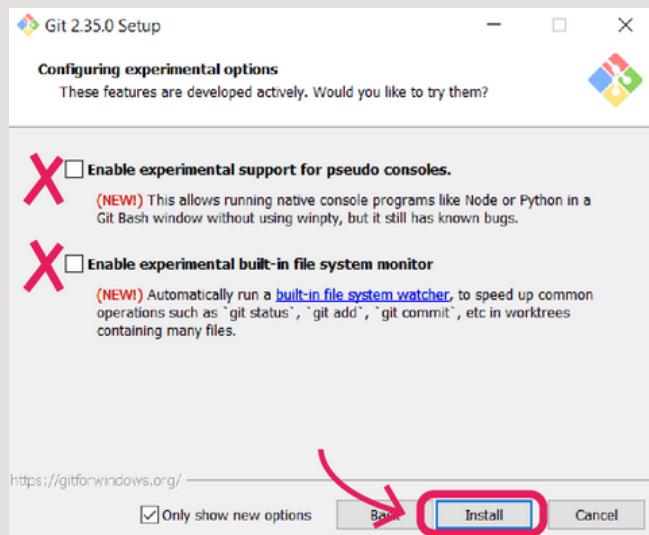
5 -Na próxima tela os recursos Git são listados para escolha de instalação. Aceite a escolha que vem por padrão e dê “Next”:



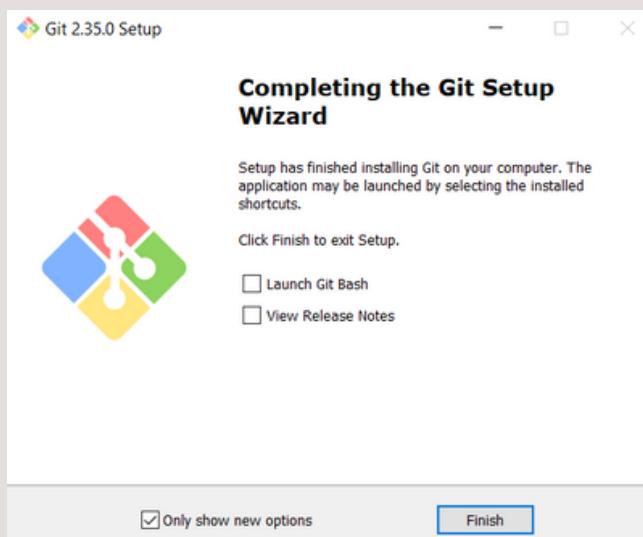
6 -Selecione “Use bundled OpenSSH” e clique em “Next”:



7 -Não selecione nenhum “experimental options” e clique em “Install”. Após clicar neste botão, o Git será instalado.



8 -Após finalizar o processo, a tela abaixo será mostrada. Finalmente, clique em “Finish” para encerrar a instalação:



Pronto! Você tem o Git instalado no seu computador!

TERMOS BÁSICOS

Quando estamos utilizando o Git para versionamento de projetos, temos alguns termos frequentemente utilizados e que é importante saber ao menos os principais deles. Vejamos alguns:



- **Pull – baixar versão mais atual**

Quando queremos trazer os arquivos do repositório remoto para **atualizar** os arquivos locais. Após o sucesso deste comando, o repositório local estará com as modificações da **versão mais atual do projeto**.

- **Commit – nova alteração com comentário**

Quando **alteramos os arquivos e criamos uma nova versão** da aplicação no repositório. Em cada commit devemos escrever uma mensagem que descreve as mudanças realizadas. É importante haver um padrão na frequência de commits e também na mensagem, para que a linha do tempo do projeto esteja sempre bem clara e documentada com as mudanças feitas.

- **Push – subir alterações**

Quando queremos levar os arquivos locais “commitados” (que já passaram pelo commit) para **atualizar o repositório remoto**. Neste caso é sempre importante garantir que o repositório local esteja atualizado (git pull), para que nenhum commit de outro membro da equipe seja perdido por sobreposição.

- **Clone – clonar**

Clonar um repositório remoto e seus arquivos para nosso local. Ou seja, esse comando **cria um repositório local com referência ao repositório remoto** clonado. O clone permite que mais de um desenvolvedor possa ter repositórios locais do mesmo projeto, subindo e baixando alterações para o mesmo repositório remoto.

- **Feature – tarefa**

É a **tarefa** que será feita. Exemplo: “trocar o tipo do menu”.



- **Branch - uma ramificação**

Uma branch consiste em uma **ramificação** do repositório remoto. Em outras palavras, quando criamos uma branch estamos replicando todo o repositório em uma “seção” independente, que será agregada à seção original posteriormente. Geralmente uma branch é criada para se implementar diferentes features, sem que o desenvolvimento de uma tenha interferência na outra. Por exemplo, dentro da Feature “trocar o tipo do menu” podemos ter branches como “deletando o menu antigo”, “criando menu novo” ou “ajustando CSS”.

- **Merge - unir ramificações em uma só**

Quando **unimos as features** desenvolvidas em branchs na ramificação original. Esse é o processo de **união de branches** do mesmo repositório, usado geralmente quando o desenvolvimento da feature na branch é finalizado.

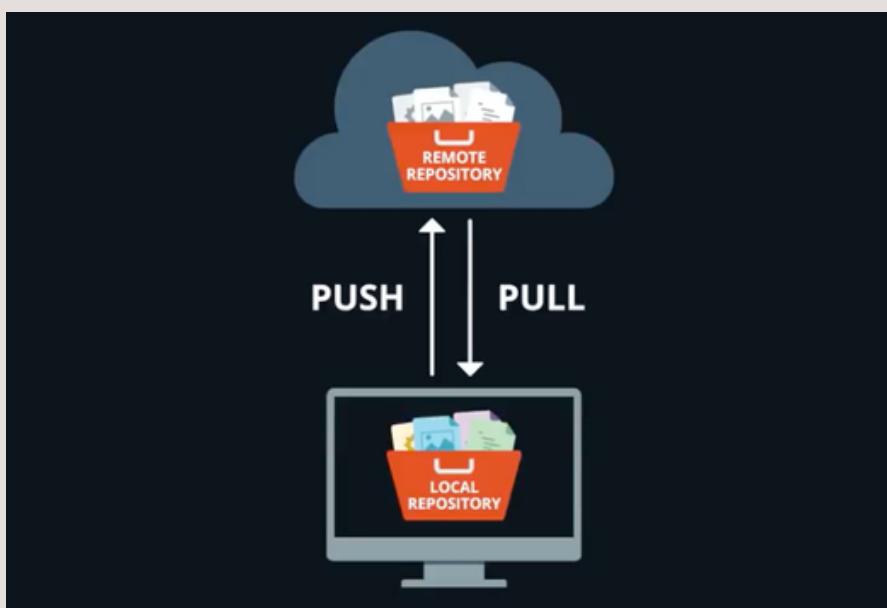
GITHUB

O GitHub é um site em que disponibilizamos um projeto versionado pelo Git de forma online. Assim podemos compartilhar o projeto com mais pessoas. O GitHub também fornece uma interface gráfica que apresenta diversos detalhes do projeto, como a versão atual e o histórico de versões, além de configurações gerais.





- Um projeto online no GitHub é definido como um **repositório remoto**. A função básica do Git e do GitHub é manter uma sincronia entre os arquivos que temos no computador (local) com os que estão online.
- Assim, todos que precisarem mexer no projeto irão acessar o site do GitHub e baixar o projeto na sua versão mais atual.



(Disponível em: [Git Remote | Learn Git](#))

No GitHub temos a escolha de definirmos um projeto **público** ou **privado**. Um **projeto privado** possui restrições de acesso. Inicialmente apenas o autor tem acesso e pode baixar o repositório para trabalhar nele. Mas o autor também pode autorizar usuários do GitHub ou um grupo (geralmente, uma equipe de desenvolvedores).

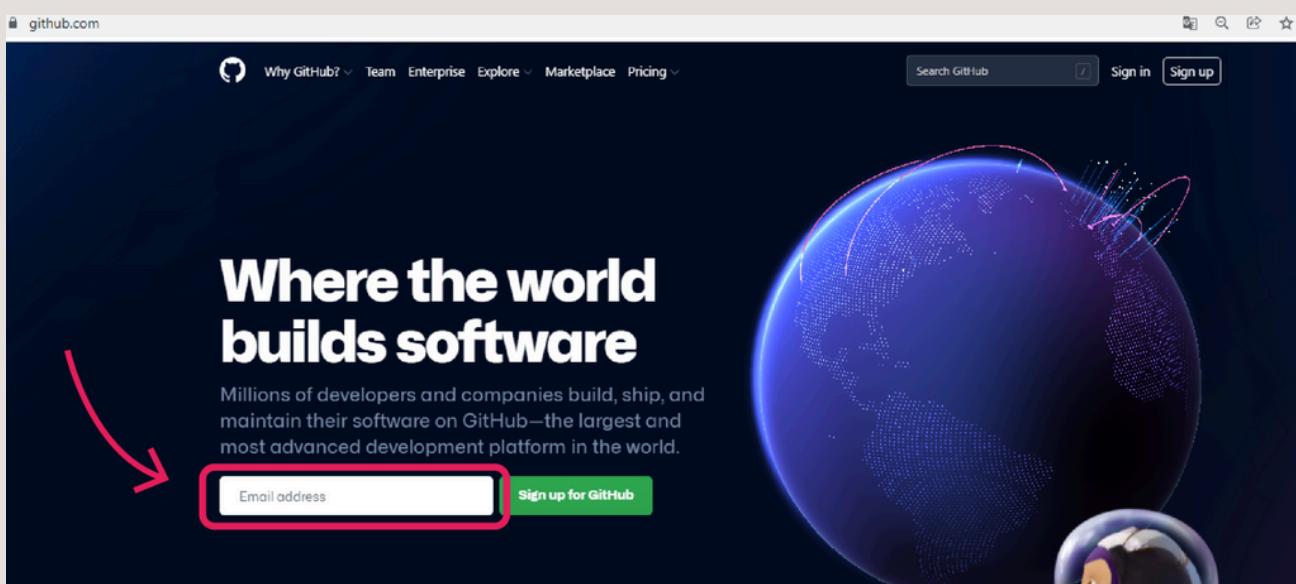
Os **projetos públicos** são abertos a todos os usuários do GitHub, ou seja, conseguimos baixar projetos de outras pessoas no nosso computador sem necessidade de permissão. Assim temos a oportunidade de estudar formas diferentes de fazer código e contribuir em diversos projetos, sejam grandes ou pequenos.



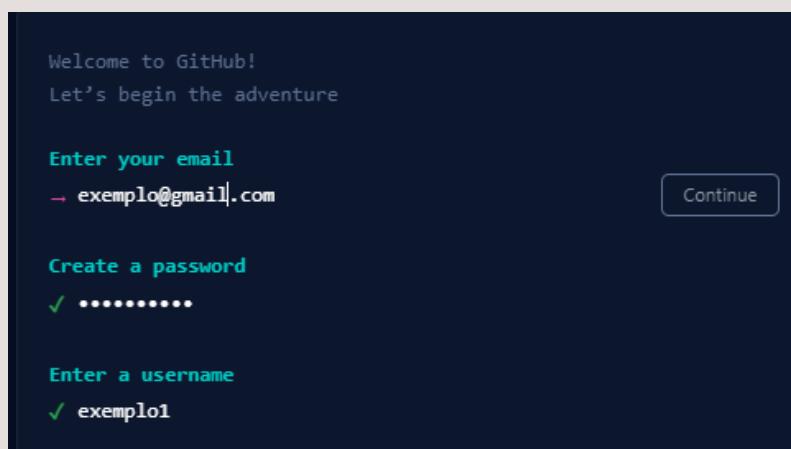
CRIANDO PERFIL NO GITHUB

Caso você já tenha conta no GitHub, pode pular esta parte. Agora vamos criar um perfil no GitHub, a partir de um e-mail que você utiliza bastante. Recomendo que seja o mesmo em que você faz login no VSCode.

1 – Insira o seu e-mail no campo no meio da tela, e clique em “Sign up for GitHub”:

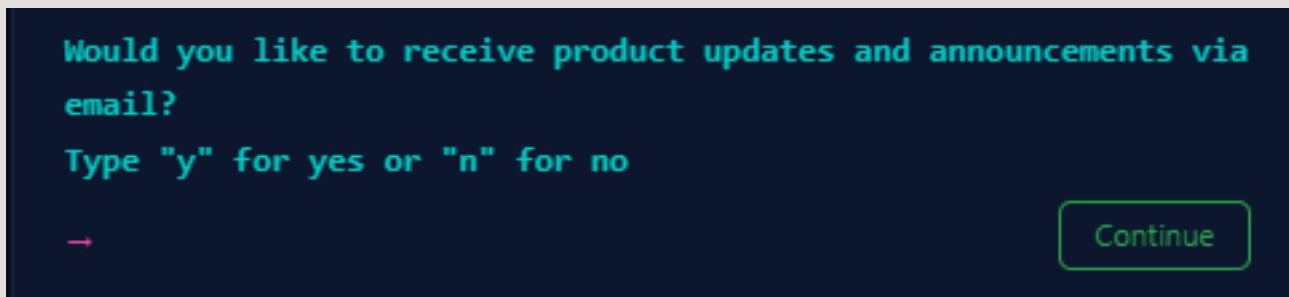


2 – Confirme o e-mail e a senha que você utilizará na próxima tela (que lembra bastante um terminal), juntamente com o nome de usuário (**username**) que você quer usar no GitHub:

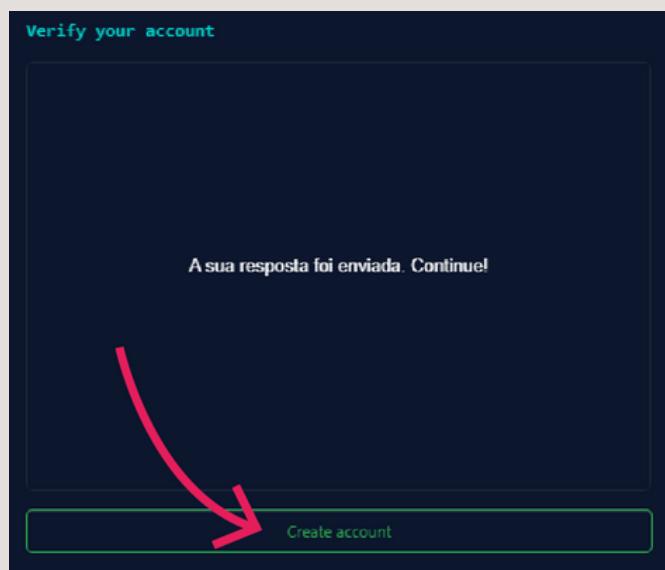




3 - A quarta opção pergunta sobre o recebimento de atualizações e anúncios do GitHub via e-mail. Escolha o que preferir, digitando **Y** para sim ou **N** para não.



4 - O próximo passo é a verificação da conta e pode aparecer de várias formas: uma imagem para você selecionar ou um áudio para você escutar e escrever o que ouviu. Faça a verificação e a tela ficará como abaixo. Clique em “Create account”:

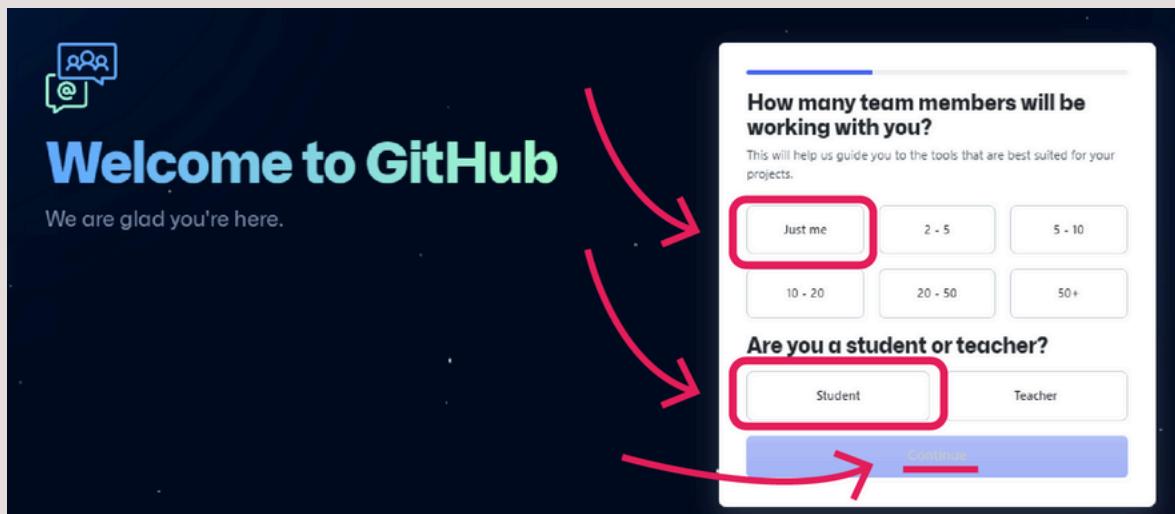


5 - Você receberá um e-mail com um código para confirmar a conta. Verifique no seu e-mail, veja o código e digite no site.

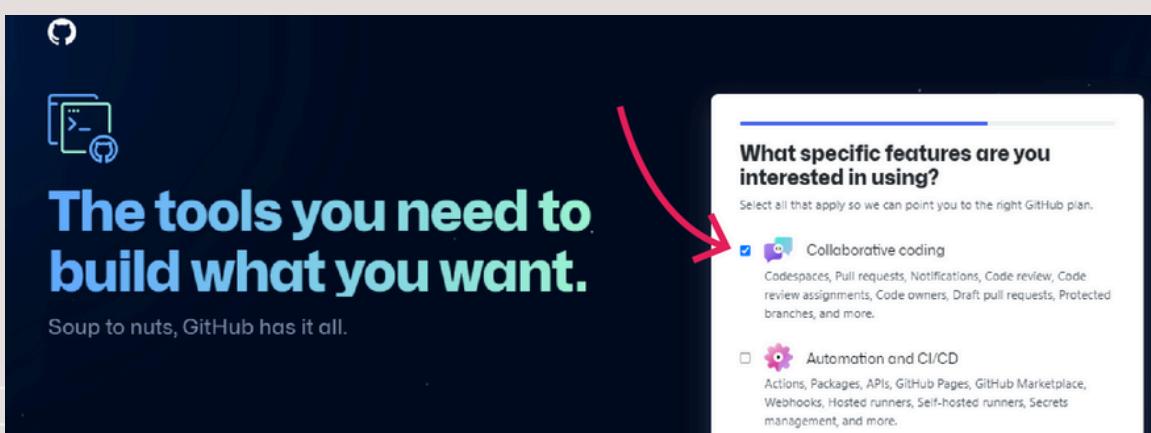




6 - Agora devemos selecionar quantas pessoas trabalham conosco e se você é professor ou estudante. Selecione “Just me” – apenas eu – e a opção “student” – estudante. Após isso, clique em “Continue”.



7 - Selecione as opções que gostaria, mas se não tiver certeza selecione a primeira focada em código colaborativo – “Collaborative coding”.



8 - Selecione o plano de estudante ou a versão free. A princípio esta parte pode ser ignorada. No caso da necessidade dos benefícios oferecidos por outros planos, o GitHub permite que seja alterado posteriormente.

Learn to ship software like a pro.

GitHub gives students free access to the best developer tools so they can learn by doing.

Free

- ① Unlimited public/private repositories
- ② 2,000 Actions minutes/month
Free for public repositories
- ③ 500MB of Packages storage
Free for public repositories
- ④ Community support

Get additional student benefits

GitHub Pro

- ⑤ Protect your branches
Ensure that collaborators on your repository cannot make irrevocable changes to branches.
- ⑥ Draft pull requests
- ⑦ Pages and Wikis
- ⑧ 3,000 CI/CD minutes/month
Free for public repositories
- ⑨ 2GB of Packages storage
Free for public repositories
- ⑩ Web-based support

GitHub Student Developer Pack

- ⑪ Free access to the industry's best developer tools
Hundreds of offers, including DigitalOcean, Microsoft Azure, Heroku, MongoDB, DataDog, Twilio, and Stripe.

GitHub Campus Expert training

- ⑫ Enrich your college technical community
Learn the skills to build diverse tech communities on campus with training, mentorships, and support from GitHub.

[Continue for free](#) [Apply for your GitHub student benefits](#)

[Skip personalization](#)

9 – Pronto, seu perfil estará criado e você será direcionado para na página inicial do GitHub:

Signed in as exemplo1

- [Set status](#)
- [Your profile](#)
- [Your repositories](#)
- [Your codespaces](#)
- [Your projects](#)
- [Your stars](#)
- [Your gists](#)

[Upgrade](#)

Feature preview

[Help](#)

[Settings](#)

[Sign out](#)

Search or jump to...

Pull requests Issues Marketplace Explore

Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

[Create repository](#) [Import repository](#)

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#) [Start a project](#)

All activity

Introduce yourself

The easiest way to introduce yourself on GitHub is by creating a README in a repository about you! You can start here:

```
exemplo1 / README.md
```

```
1 - 🚀 Hi, I'm @exemplo1
2 - 🌐 I'm interested in ...
3 - 🎓 I'm currently learning ...
4 - 🛠️ I'm looking to collaborate on ...
5 - 📧 How to reach me ...
6 -
```

[Dismiss this](#) [Continue](#)



ORIENTAÇÃO PARA PORTFÓLIO

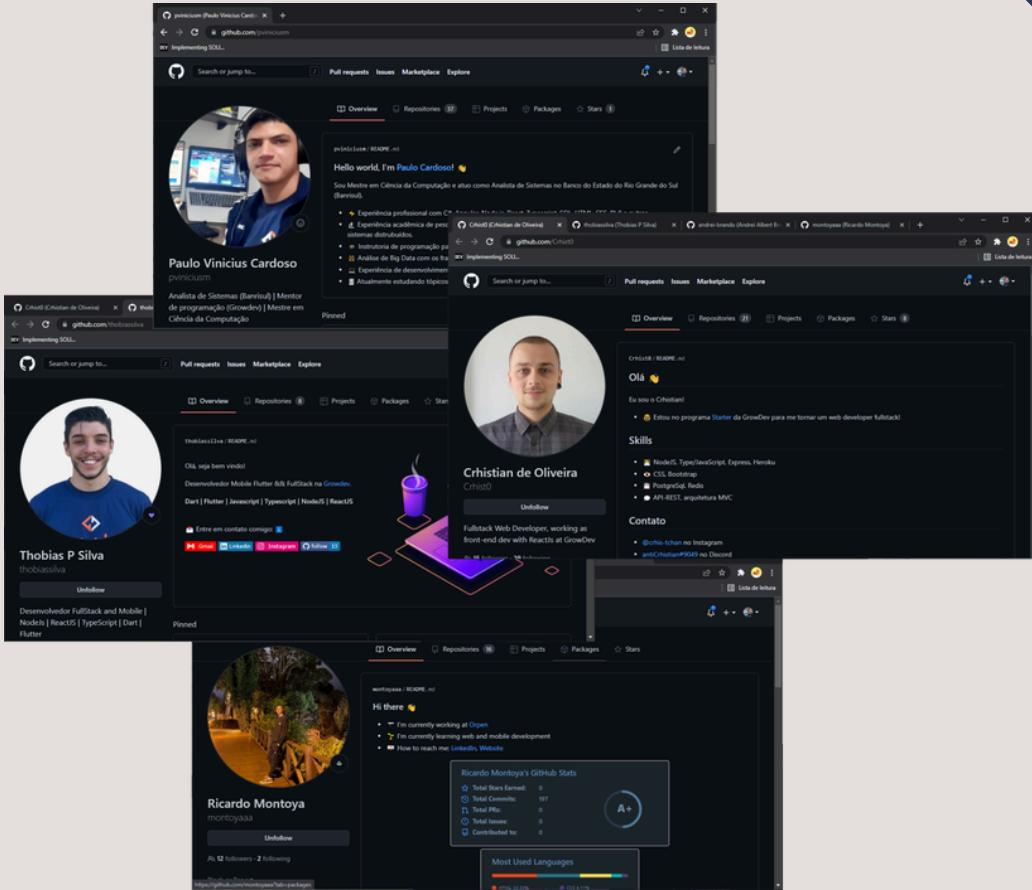
Estima-se que o GitHub tem atualmente mais de **73 milhões** de usuários ativos na sua plataforma, com mais de **200 milhões** de repositórios públicos e privados. Esses dados mostram como a popularização da plataforma está em evidência.

Por isso, hoje o GitHub serve como um **repositório dos projetos** de cada usuário e tem um grau de confiabilidade muito grande. Todos os códigos feitos e a linha do tempo mostram de que forma cada usuário trabalhou até chegar na conclusão de cada aplicação. Recrutadores têm acesso às principais linguagens usadas, às contribuições em projetos de outros usuários e diversas outras estatísticas.

Portanto é possível afirmar de forma assertiva que **ter um bom portfólio no GitHub é essencial para um candidato na área de programação**. Lembre-se de manter uma boa organização de projetos, de realizar commits eficazes e, se possível, contribua com outros projetos. Todos esses aspectos contam muito para o seu currículo!

Outro ponto muito importante no GitHub é o **readme inicial**: nele podemos fazer um resumo do perfil para nossa apresentação e para descrever os principais projetos e skills.

Fonte: <https://github.com/about>



Caso queira aprender mais sobre como fazer um readme, além da organização de aplicações e commits no GitHub, os links abaixo podem ser bastante úteis:

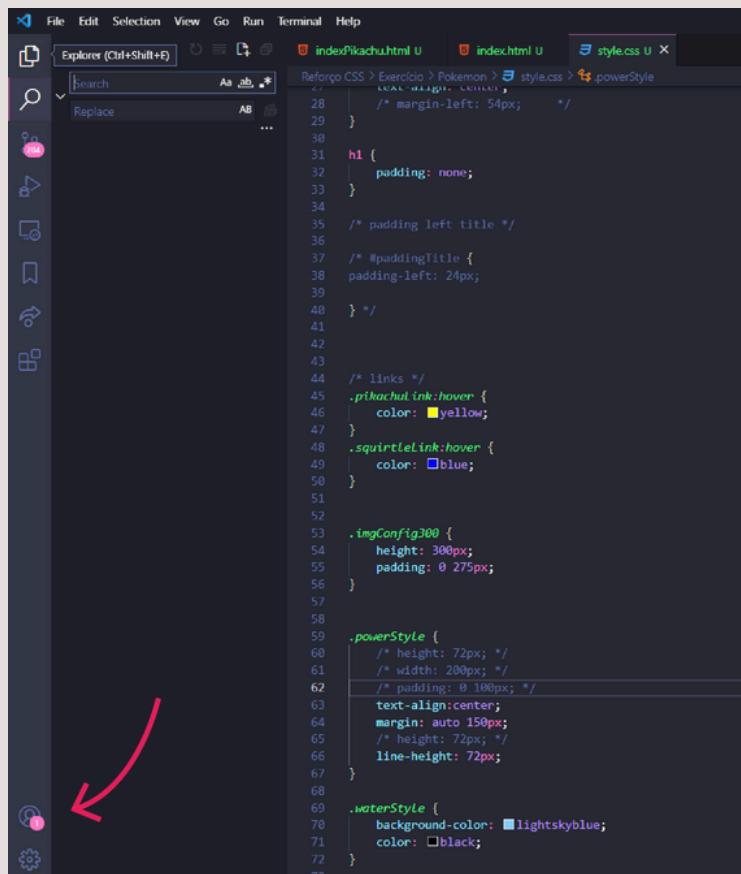
- [Como escrever uma boa mensagem de commit](#)
- [Como escrever boas mensagens de commit: um guia prático do Git](#)
- [Antes e depois das minhas mensagens de commit | by Ingrid Mendes | Code Like A Girl](#)
- [Como criar um README incrível para seus projetos \(e perfil\) no Github](#)
- [Sintaxe básica de escrita e formatação no GitHub](#)
- [Criando um README para o seu perfil no GitHub - DEV Community](#)



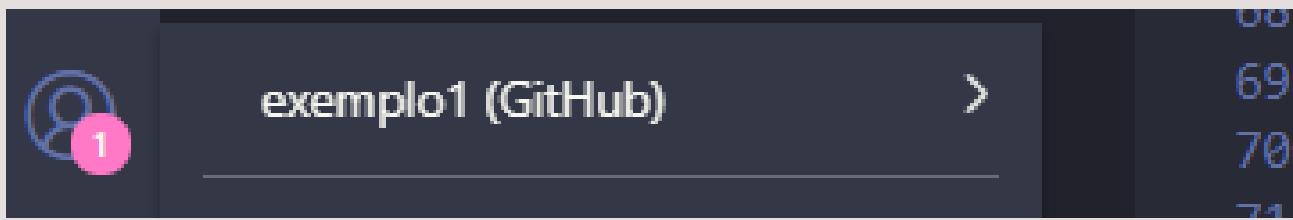
UTILIZANDO GIT NO VS CODE

Até certo tempo atrás, o uso mais comum do git era através de comandos de linha, onde se digitava o que queria fazer em um terminal. Hoje, a IDE Visual Studio Code (**vsCode**) possui um **plugin** que atende bem a integração com o Git e com o GitHub, de modo que processo se torna mais rápido. A seguir, é descrito o passo-a-passo de integração do VSCode com o GitHub.

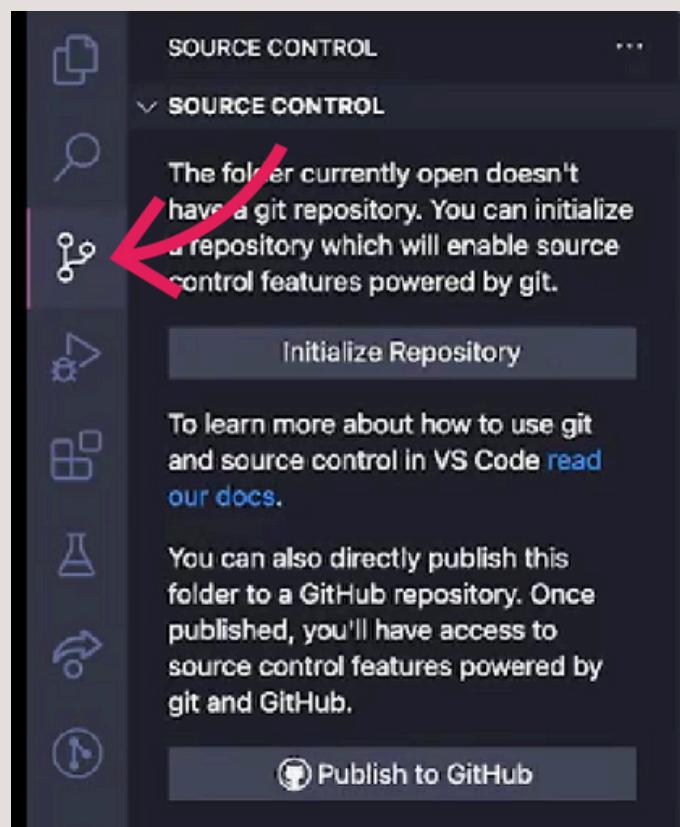
1- Para ter acesso à conta do GitHub, faça login pelo VS Code, selecionando o ícone de perfil do lado esquerdo do programa:



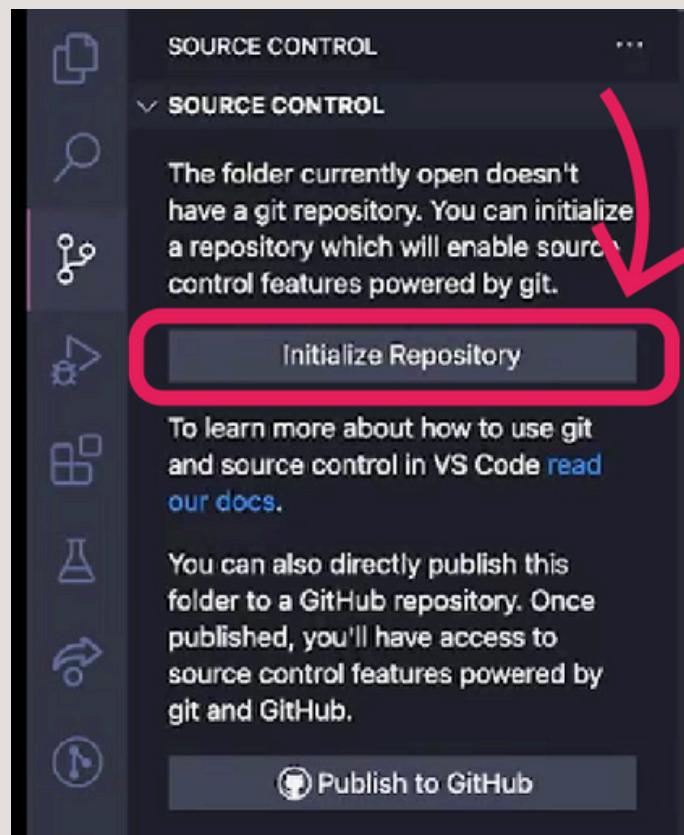
2 - Selecione a opção de login - sign in, e utilize e-mail e senha cadastrados no GitHub. Provavelmente ele irá redirecionar você para o site do GitHub, perguntando se você aceita utilizar este usuário no VSCode, assim **confirme e abra o VSCode**. Verifique se o usuário agora está com o login no VS Code:



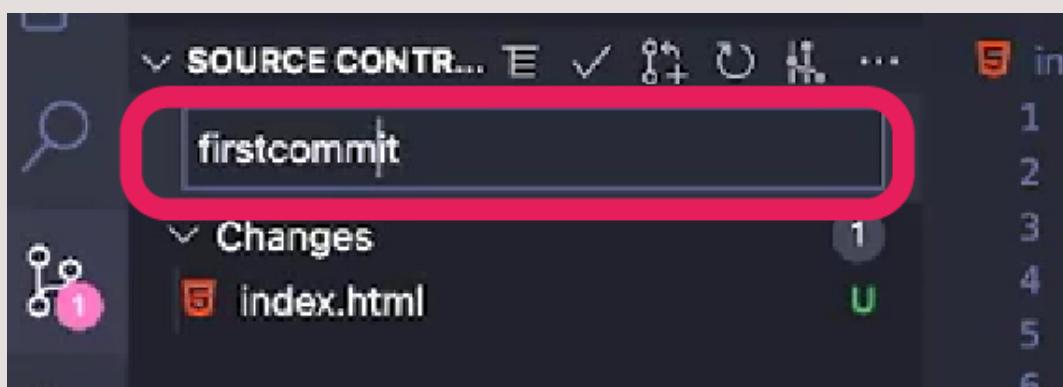
3 – Abra seus arquivos, ou pasta de projeto que você deseja salvar no GitHub. Com a pasta e arquivos prontos, clique no ícone do git, do lado esquerdo do VSCode:



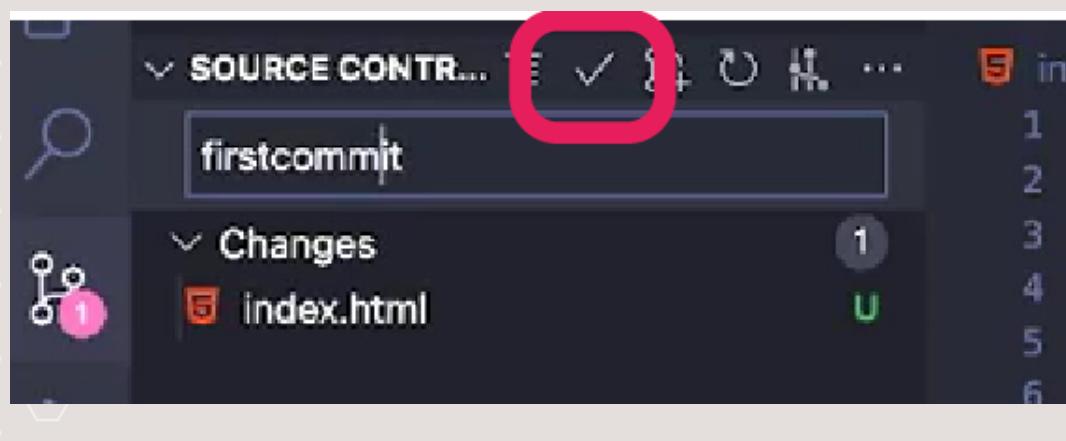
4 – Selecione a opção “**Initialize Repository**” – initialize o repositório – para que ele crie o repositório na pasta que você está trabalhando.



5 – Escreva o commit que deseja no campo de texto:

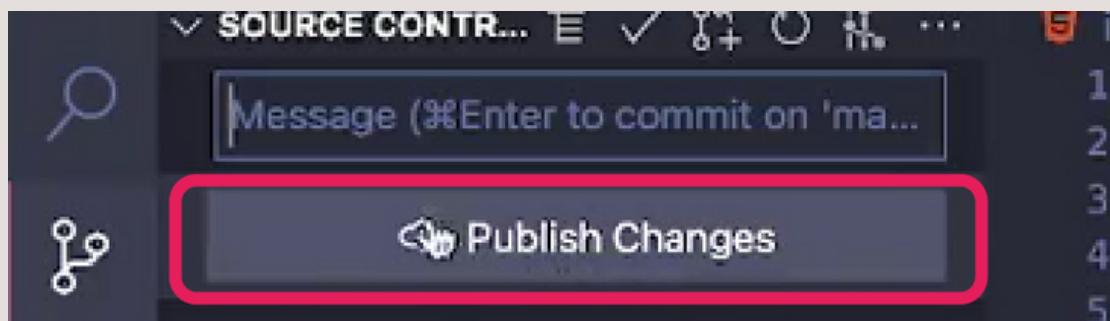


6 – Após escrever, selecione o símbolo de ok para confirmar.

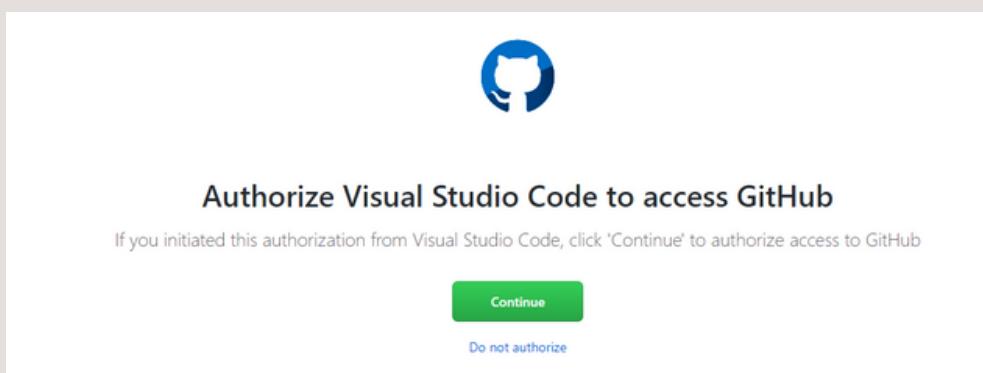




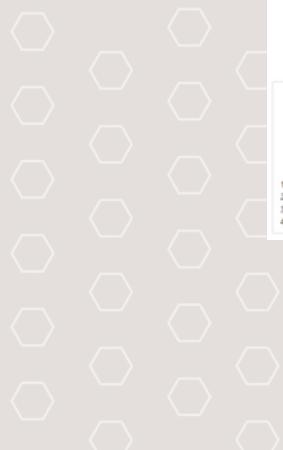
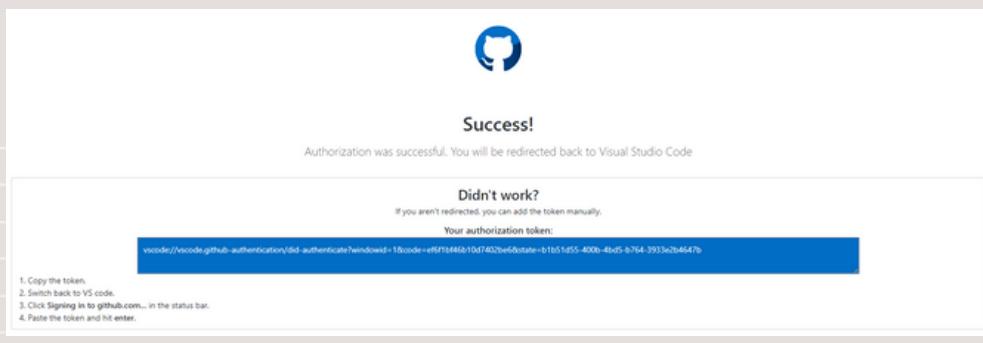
7 – Agora vamos publicar, clique em “Publish Changes” ou “Publish Branch”



8 – Após isso, pode aparecer mais uma vez a janela de confirmação do GitHub pedindo autorização para utilização com o VS Code. Confirme.



9 – Após fazer as confirmações necessárias, talvez ele mostre a opção de abrir novamente no VSCode. Esta tela confirmará que deu certo.

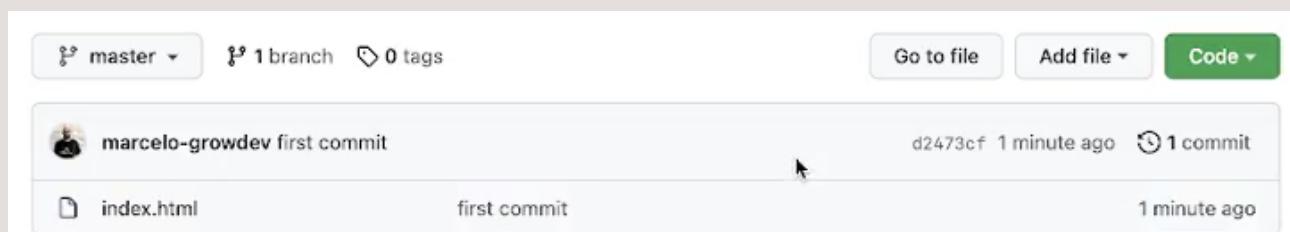




10 - Ao retornar ao VS Code com esta parte ajustada, e clicando novamente em Publish, aparecerá duas opções: repositório private – privado – e public – público. Selecione uma das duas opções:



11 - Após finalizar, entre no site do GitHub, e ao clicar no perfil, selecione “**Repositories**” – repositórios – e verifique se está tudo ok.



ALTERNATIVAS AO GITHUB

Apesar da grande popularidade do GitHub, existem outras opções quando o assunto é plataforma de hospedagem para repositórios remotos. As buscas por alternativas se intensificaram após a compra da plataforma pela Microsoft, em 2018. Hoje as alternativas mais populares são o GitLab e o BitBucket.

GitLab

A alternativa mais usada é o GitLab, outra plataforma de versionamento de código que possui uma proximidade muito grande com o GitHub em termos de facilidade de uso. Uma importante característica do GitLab é que a plataforma é de código aberto. Ou seja, é possível que os próprios programadores contribuam com o projeto.



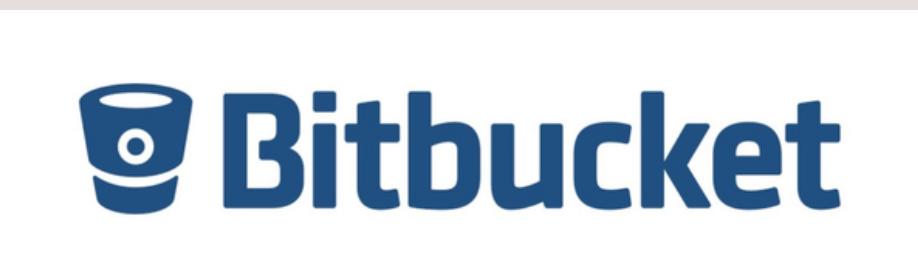
GitLab



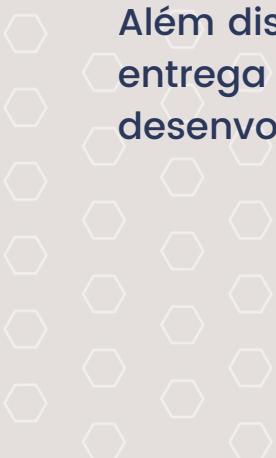
O GitLab é usado por diversas instituições importantes, dentre elas o Instituto de Pesquisa da Marinha do Brasil (IPqM), o SpaceX e até mesmo a NASA. Além do repositório de código, o GitLab possui serviços de documentação de projetos (wiki), gerenciamento de tarefas e também fornece ferramentas para automatização de entregas (CI/CD).

BitBucket

O BitBucket é um sistema para versionamento de código que faz parte do conjunto de aplicações da Atlassian. Assim uma das grandes vantagens desta plataforma é a integração com o Trello – gerenciador de tarefas – e com o popular gerenciador de projetos Jira.



O BitBucket oferece um número ilimitado de repositórios remotos para os usuários com um máximo de 3 colaboradores por projeto. Além disso, é possível encontrar recursos de DevOps (integração e entrega contínua) e diversos facilitadores para equipes de desenvolvedores.





SITES QUE SALVAM

Durante o desenvolvimento com o Git e Github, diversos problemas podem ocorrer e com isso, você terá que rodar comandos além dos listados em aula.

Para isso, indicamos 2 sites :

Oh SHIT GIT

Oh Shit, Git! é uma ferramenta online criada para ajudar desenvolvedores a resolver problemas comuns que podem ocorrer ao usar o Git, um sistema de controle de versão. Ele oferece uma coleção de soluções para situações comuns em que você pode se encontrar, como reverter mudanças, resolver conflitos e desfazer commits. A ideia é fornecer respostas rápidas e práticas para problemas frequentes que podem ser desafiadores para quem está começando ou até para usuários mais experientes. O site é projetado para ser uma referência rápida, com dicas e comandos úteis para manter seu repositório Git em ordem.

https://ohshitgit.com/pt_BR

GIT Guia Prático sem complicaçāo

Git - Guia para Iniciantes, é um guia de introdução ao Git em português. Ele é voltado para pessoas que estão começando a usar o sistema de controle de versão Git. O guia cobre conceitos básicos e comandos essenciais de forma clara e direta.

https://rogerdudler.github.io/git-guide/index.pt_BR.html

REFERÊNCIAS

1. [Git e GitHub: quais as diferenças? | Blog TreinaWeb](#)
2. [Git e Github: O que são, Como Configurar e Primeiros Passos | Alura Cursos Online](#)
3. [O que é Git e GitHub, para que serve e quais as vantagens?](#)
4. [Git // Dicionário do Programador](#)
5. [GITHUB // Dicionário do Programador](#)
6. [O QUE É GIT E GITHUB? – definição e conceitos importantes 1/2](#)
7. [Usar as ferramentas de controle de versão do Git no Visual Studio Code - Learn | Microsoft Docs](#)
8. [Git Flow // Dicionário do Programador](#)
9. [Git Flow: a estratégia essencial para organizar as versões de um código](#)
10. [Git: O que é e como funciona? Seja um profissional do Git! | Le Wagon](#)
11. [Git, GitHub e GitLab o que é cada um deles? – Diolinux](#)
12. [GitLab vs GitHub: Explore suas Principais Diferenças e Semelhanças](#)
13. [Difference Between Bitbucket and GitHub - GeeksforGeeks](#)
14. [Bitbucket: entenda como funciona este serviço de hospedagem de projetos](#)
15. [Oh Shit, GIT](#)
16. [Git Guia Prático Sem Complicação](#)

BONS ESTUDOS!

academy
by growdev



growdev.com.br