4️⃣

Módulo 4 - Gerenciando seu código com maestria

Fala, **dev**!

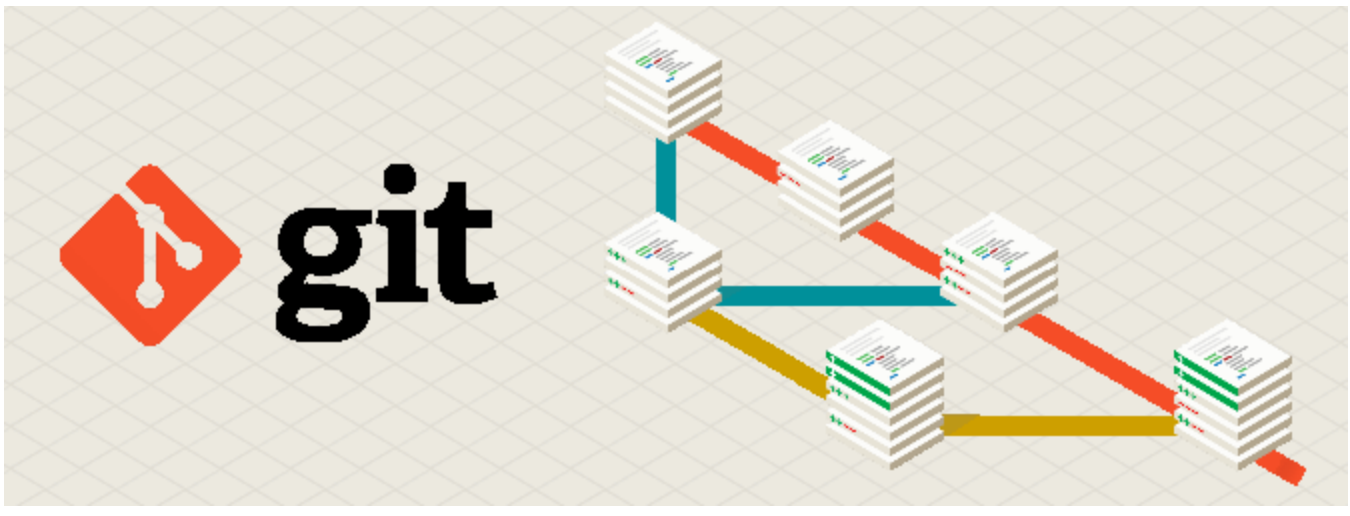
Neste módulo, vamos aprender a utilizar uma ferramenta essencial para um programador: o Git. Além disso, colocaremos o nosso código no GitHub, a maior plataforma de repositório de códigos online. Vamos nessa?

O Surgimento do Git - A História por Trás

Imagine um cenário: você é um programador talentoso, trabalhando em um projeto de software incrível com outras pessoas espalhadas pelo mundo. Todos vocês precisam colaborar, compartilhar código e manter o histórico das mudanças. Até certo ponto, isso é fácil. Vocês enviam e-mails com arquivos anexos, mas logo se torna um pesadelo. Arquivos perdidos, versões conflitantes, caixas de entrada lotadas.

Foi aí que **Linus Torvalds**, o criador do sistema operacional Linux, teve uma ideia brilhante. Ele desenvolveu o Git. O Git é um sistema de controle de versão distribuído. Isso significa que cada programador em um projeto tem uma cópia completa do histórico de todo o projeto, o que é genial.

E por que o Git é importante para um programador?



1. **Controle de Versão:** O Git permite que você acompanhe todas as mudanças feitas em seu código ao longo do tempo. Você pode ver quem fez o quê, quando e por quê. Isso é útil quando algo dá errado e você precisa voltar no tempo para encontrar o erro.
2. **Colaboração Simples:** Com o Git, vários programadores podem trabalhar no mesmo projeto ao mesmo tempo. Você pode fazer alterações em seu código e, em seguida, mesclá-las com as alterações de outra pessoa sem problemas.

Isso torna a colaboração mais fácil.

3. **Armazenamento na Nuvem:** Serviços como GitHub e GitLab oferecem espaço gratuito para hospedar seus projetos Git na nuvem. Isso significa que você pode acessar seu código de qualquer lugar e compartilhá-lo facilmente com outros programadores.
4. **Ramificações (Branches):** Com o Git, você pode criar ramificações para experimentar novos recursos ou corrigir bugs sem afetar o código principal. Isso é ótimo para experimentação e desenvolvimento seguro.
5. **Segurança:** Como cada programador tem uma cópia completa do histórico do projeto, ele é mais resistente a perdas de dados. Mesmo se um servidor quebrar, o histórico do projeto ainda estará seguro em muitos outros lugares.

O que é código aberto?

Código aberto é como uma receita de bolo que todo mundo pode ver e usar. Imagine que você está fazendo um bolo delicioso e, em vez de manter a receita em segredo, você a coloca na internet para que qualquer pessoa possa ver, usar e até mesmo melhorar.

Da mesma forma, o **código aberto** é como se os programadores compartilhassem suas receitas de software. Isso significa que outras pessoas podem dar uma olhada, fazer melhorias e até mesmo criar novos pratos a partir da mesma receita.

Essa ideia de compartilhar e colaborar faz com que o software fique cada vez melhor, porque muitas pessoas diferentes podem contribuir com suas ideias e correções. Portanto, quando você ouve falar de um software de código aberto, pense nele como uma receita de bolo que está disponível para todos, e todos podem ajudar a torná-lo mais saboroso! 😊🍰

GitHub: A rede social dos desenvolvedores

Você já ouviu falar do GitHub? Ele é como uma rede social para desenvolvedores de software, um lugar onde você não só compartilha seu código, mas também colabora com outros desenvolvedores em projetos incríveis. Vale dizer que existem outras plataformas além do GitHub, como o GitLab e o Bitbucket.

Agora, vamos explorar o que torna o GitHub tão especial e entender como ele difere do Git, a ferramenta que todos os desenvolvedores adoram.

A Origem

O GitHub surgiu em 2008 como uma plataforma para compartilhar e colaborar em projetos de software. Imagine isso como uma rede social, mas para programadores. Seu fundador, Tom Preston-Werner, queria tornar a colaboração em código mais fácil.

Hoje em dia, o GitHub é como um playground para desenvolvedores. Eles podem compartilhar seu código, trabalhar juntos em projetos e controlar as mudanças que fazem. É como um superclube de programadores, onde todos podem aprender e criar juntos.

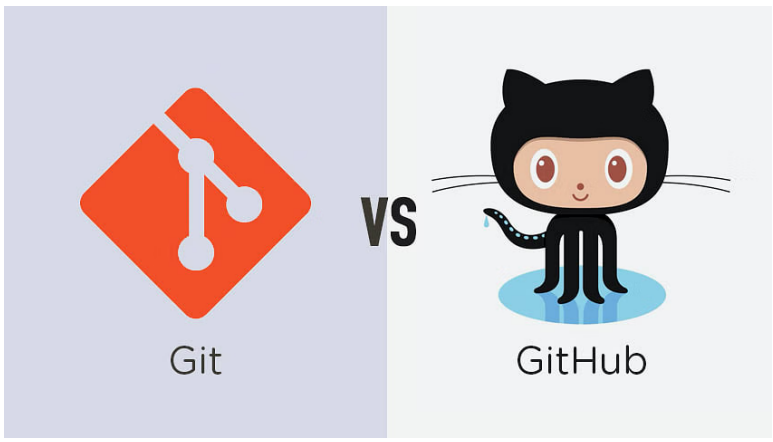
É um lugar onde a magia da programação acontece, onde ideias se transformam em software e onde pessoas de todo o mundo se conectam por meio do código. Uma história incrível de como uma ideia simples se tornou um dos lugares mais importantes para os desenvolvedores.

Em 2018, a Microsoft viu o potencial do GitHub e o adquiriu. Mas não se preocupe, ele ainda é independente e comprometido com o código aberto.

Git e GitHub: Qual é a diferença?

Às vezes, as pessoas confundem o Git com o GitHub. É como confundir um arquivo de vídeo no seu computador com o YouTube.

O YouTube é como um lugar onde você pode levar um vídeo do seu computador e compartilhá-lo com o mundo. Ele hospeda seus vídeos, permite que outras pessoas assistam e comentem.



Da mesma forma, o GitHub é como um "YouTube" para os projetos de programação, mas não para vídeos, para os arquivos de código (chamados de repositórios Git). É como se você estivesse compartilhando uma pasta com todos os arquivos do seu projeto. No GitHub, outras pessoas podem acessar e baixar essa pasta, deixar comentários e até mesmo contribuir com melhorias no seu projeto.

Assim como o YouTube tem canais para organizar vídeos, o GitHub possui contas. Cada conta no GitHub é como a sua vitrine, onde você pode mostrar seus projetos, adicionar informações sobre si mesmo e interagir com outros programadores.

O GitHub é uma comunidade de programadores onde você pode compartilhar e colaborar em projetos. O primeiro passo é criar uma conta gratuita em github.com.

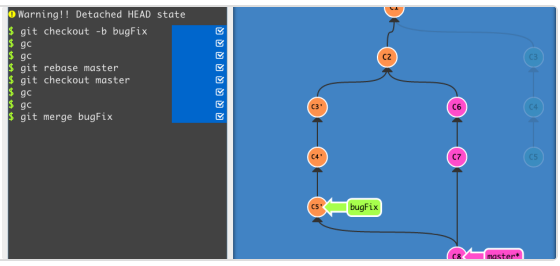
Resumindo, o Git é como o seu arquivo de vídeo e o GitHub é como o YouTube para projetos de programação. 🚀

▼ Links importantes

Learn Git Branching

An interactive Git visualization tool to educate and challenge!

https://learngitbranching.js.org/?locale=pt_BR



10 comandos do Git que todo desenvolvedor deveria conhecer

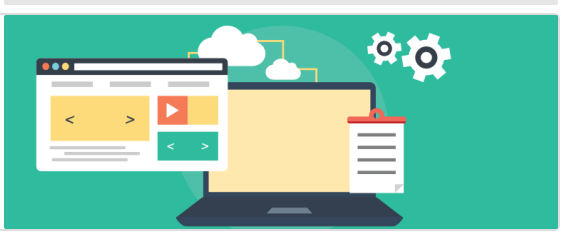
O Git é uma parte importante da programação no dia a dia (especialmente se você estiver trabalhando em equipe) e é amplamente usado no setor de software. Como existem muitos comandos que você pode utilizar, dominar o Git por completo leva tempo. Alguns comandos, no

https://www.freecodecamp.org/portuguese/news/10-comandos-do-git-que-todo-desenvolvedor-deveria-conhecer/

Comandos Básicos de GIT

Mais informações sobre GIT básicos podem ser recuperados aqui. Neste tutorial, os comandos git mais básicos serão falados.

http://hostinger.com.br/tutoriais/comandos-basicos-de-git



Top 25 comandos do Git

Controle de Versão, Controle de Revisão, Controle de Origem, ou os termos originais em inglês Version Control, Source Control, tanto faz o nome. Esse processo lida com o gerenciamento de alterações em documentos, programas de computador, sites da Web ou apenas sobre qualquer

https://www.codigofonte.com.br/artigos/top-25-comandos-do-git

▼ Como instalar o Linux dentro do Windows usando WSL?

> ### CURADORIA DE CONTEÚDO

Eu separei alguns vídeos públicos no YouTube que podem ser úteis para te ajudar na instalação do Linux dentro do seu sistema Windows.

Essa instalação pode ser útil para que você tenha mais interação com o Linux dentro do mesmo ambiente Windows e possa utilizar os comandos do terminal Linux sem muitos problemas.

WSL 2 - A solução para rodar Linux dentro do Windows 10 - Root #08

✅ Inscreva-se na NLW Heat: https://bit.ly/3FRg4Nx

Vamos instalar o WSL2, subsistema do Windows para linux, permitindo que seja executado um

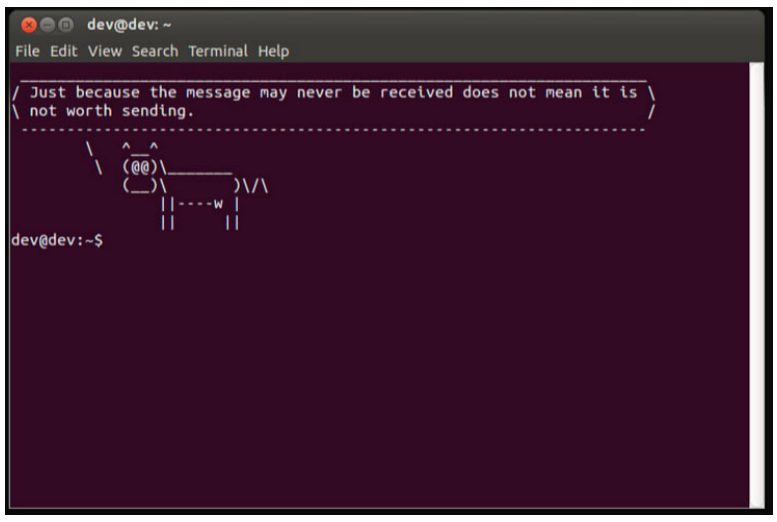
https://www.youtube.com/watch?v=hd6lxt5iVsg&ab_channel=Rocketseat

Como instalar o Linux no Windows com WSL

🌐 Descubra o Caminho Mais Curto para se tornar um Programador! Conheça a Comunidade DevPro: https://l.dev.pro.br/comunidade-dev-pro-yt

https://www.youtube.com/watch?v=MaTe1qBTaic&ab_channel=CanalDevPro

Quais são os principais comandos do terminal Linux?



▼ Comandos Básicos do Terminal Bash

O terminal Bash é uma ferramenta poderosa para interagir com o sistema operacional Linux. Vamos aprender alguns comandos básicos:

🖥️ Comando `pwd` - Mostrar Diretório Atual

Para descobrir em qual diretório você está no momento, use o comando `pwd` (Print Working Directory):

```
pwd
```

🖥️ Comando `ls` - Listar Arquivos e Diretórios

O comando `ls` é usado para listar o conteúdo de um diretório:

```
ls
```

- Opção `ls -l` ou `ll` - Lista Detalhada

Para obter uma lista detalhada dos arquivos e diretórios, use `ls -l`:

```
ls -l
```

🖥️ Comando `cd` - Navegar entre Diretórios

O comando `cd` é usado para entrar em diretórios:

```
cd nome_do_diretorio
```

🖥️ Comando `rm` - Remover Arquivos e Diretórios

O comando `rm` no terminal Linux é usado para remover arquivos e diretórios. No entanto, é essencial usá-lo com cuidado, pois a exclusão é permanente e não pode ser desfeita.

```
rm arquivo.txt          # Remove um arquivo
rm -r diretorio         # Remove um diretório e seu conteúdo (recursivamente)
```

- **Remover Arquivos:**

  Para remover um arquivo específico, você pode usar o `rm` seguido do nome do arquivo. Por exemplo:

  ```
  rm arquivo.txt
  ```

- **Remover Diretórios Vazios:**

  Para remover um diretório vazio (sem arquivos dentro), use o `rm` com a opção `-d`. Por exemplo:

  ```
  rm -d diretorio_vazio/
  ```

- **Remover Diretórios com Conteúdo:**

  Para remover um diretório e seu conteúdo, você deve usar o `rm` com a opção `-r` (*recurse*). Por exemplo:

  ```
  rm -r diretorio_com_conteudo/
  ```

- **Remover Diretórios e seu Conteúdo de Forma Forçada:**

  Às vezes, você pode encontrar diretórios ou arquivos protegidos contra exclusão e, nesse caso, você pode usar a opção `-f` (force) para forçar a remoção. No entanto, tenha muito cuidado ao usar `-f`, pois ele não pedirá confirmação e excluirá tudo sem perguntar. Por exemplo:

  ```
  rm -rf diretorio_com_conteudo_protegido/
  ```

Tenha em mente que a exclusão é uma operação irreversível, portanto, use esses comandos com extrema precaução e certifique-se de que você está excluindo os arquivos e diretórios corretos. Verifique duas vezes antes de pressionar Enter, especialmente ao usar as
opções `-r` ou `-f`.

🖥️ Comando `mkdir` - Criar Diretório

O comando `mkdir` é usado para criar um novo diretório:

```
mkdir nome_do_diretorio
```

🖥️ Comando `cp` - Copiar Arquivos e Diretórios

O comando `cp` é usado para copiar arquivos e diretórios:

```
cp arquivo.txt destino/  # Copia um arquivo para um diretório
cp -r diretorio/ destino/ # Copia um diretório e seu conteúdo para outro local
```

🖥️ Comando `mv` - Mover/Renomear Arquivos e Diretórios

O comando `mv` é usado para mover ou renomear arquivos e diretórios:

```
mv arquivo.txt novo_nome.txt       # Renomeia um arquivo
mv arquivo.txt destino/            # Move um arquivo para outro local
mv arquivo.txt pasta/novo_nome.txt # Move um arquivo para outro local e renomeia
```

🖥️ Comando `touch` - Criar Arquivos Vazios

O comando `touch` é usado para criar arquivos vazios:

```
touch novo_arquivo.txt
```

🖥️ Comando `cat` - Exibir Conteúdo de Arquivos

O comando `cat` é usado para exibir o conteúdo de arquivos:

```
cat arquivo.txt
```

🖥️ Comando `cat >` - Criar ou Sobrescrever Arquivos

O comando `cat >` é usado para criar ou sobrescrever o conteúdo de arquivos:

```
cat > novo_arquivo.txt
```

Digite o conteúdo e pressione `Ctrl + D` para salvar ou `Ctrl + C` para encerrar.

▼ Referências

Iniciando o uso do terminal do linux

Uma das coisas que mais assusta pessoas que não conhecem ou estão iniciando na área de computação é o uso do terminal em um sistema Linux. Eu, assim como muitos, relutei muito para começar a utilizar, até o momento que não tive saída, mas me surpreendi muito, não é que

https://www.ufsm.br/pet/sistemas-de-informacao/2020/04/29/iniciando-o-uso-do-terminal-do-linu

40 Comandos Linux Que Todo Usuário Deve Conhecer

Que tal 40 Comandos Linux para você aprender e conseguir usar o sistema operacional com muito mais propriedade? Aqui tem!

https://www.hostinger.com.br/tutoriais/comandos-linux

▼ Editor de texto no terminal

Vim

O `vim` é um editor de texto avançado que é muito poderoso, mas também pode parecer complexo para iniciantes. Aqui estão os conceitos básicos:

- **Abrir um arquivo com o Vim:**
  - Digite o seguinte comando no terminal:

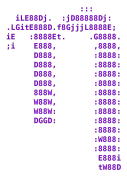
```
vim nome_do_arquivo
```

Pressione a tecla `Enter`.

- **Navegar no Vim:**
  - Use as teclas de seta ou as teclas `h` (esquerda), `j` (baixo), `k` (cima) e `l` (direita) para se mover pelo texto.
- **Inserir texto:**
  - Para começar a inserir texto, pressione a tecla `i`. Você verá "-- INSERT --" na parte inferior.
  - Agora você pode digitar o texto normalmente.
- **Salvar e sair do Vim:**
  - Pressione a tecla `Esc` para sair do modo de inserção.
  - Digite `:w` e pressione `Enter` para salvar o arquivo.
  - Digite `:q` e pressione `Enter` para sair do Vim.
- **Salvar e sair de uma vez:**
  - Para salvar e sair em um único comando, você pode digitar `:wq` e pressionar `Enter`.

Nano

O `nano` é um editor de texto mais simples e amigável para iniciantes. Aqui estão os conceitos básicos:



- **Abrir um arquivo com o Nano:**
  - Digite o seguinte comando no terminal:

    ```
    nano nome_do_arquivo # Exemplo: teste.txt
    ```

    Pressione a tecla `Enter`.
- **Navegar no Nano:**
  - Use as teclas de seta para se mover pelo texto.
- **Inserir texto:**
  - Basta começar a digitar. Não há necessidade de entrar em um modo especial.
- **Salvar e sair do Nano:**
  - Pressione `Ctrl` + `O` para salvar o arquivo.
  - Ele pedirá que você confirme o nome do arquivo. Pressione `Enter`.
  - Para sair, pressione `Ctrl` + `X`.

GitHub: Os primeiros passos

Se você quer ser um programador de verdade, então precisa ter uma conta no GitHub.

Por que? Porque o GitHub é a maior plataforma de hospedagem de código-fonte e arquivos usando controle de versão Git.

Isso significa que muitas pessoas e empresas utilizam o GitHub para manter os seus projetos e além disso, você pode pesquisar diversos repositórios de código aberto que podem te ensinar sobre algo e ajudar no seu projeto de software.

Agora, que você sabe disso, vamos criar a sua conta!

Clicando no link abaixo, você vai cair direto na página de cadastro 👇🏻

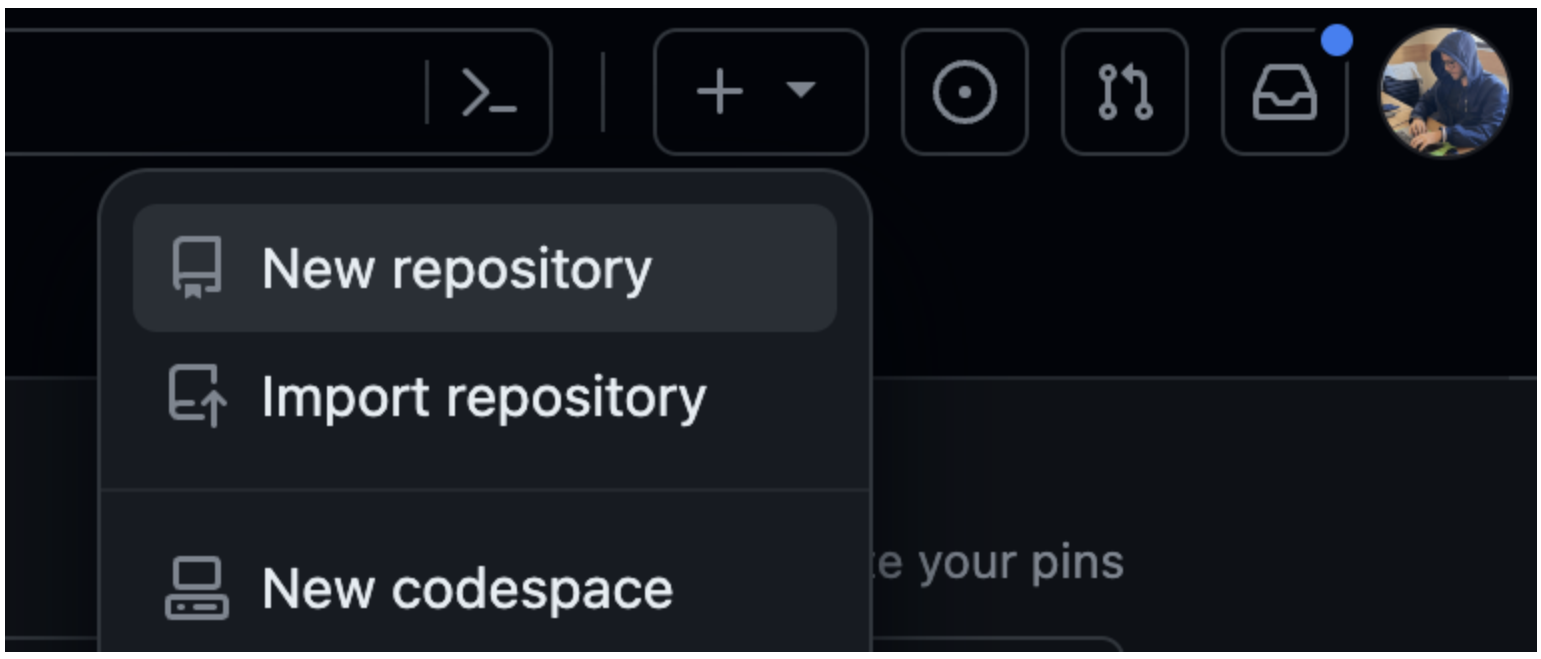
GitHub

GitHub is where people build software. More than 100 million people use GitHub to discover, fork, and contribute to over 330 million projects.

https://github.com/signup

Criando um repositório no GitHub

Quando você estiver logado na sua conta do GitHub, você irá perceber um ícone de ✚ bem no canto superior direito, igual a imagem abaixo:



Basta você clicar em *New repository* e você poderá preencher os dados relacionados ao seu repositório. Eu explico isso durante a aula com título “**Criando nosso repositório remoto no GitHub**”, se tiver alguma dúvida, basta assistir.

Gerando uma chave SSH

Para que você possa interagir com seu repositório remoto dentro do GitHub usando o Git, você precisará configurar a sua chave SSH.

Essa chave será responsável por identificar o seu computador como “autorizado para subir atualizações” para sua conta no GitHub. Ou seja, a chave SSH vai ser um “crachá de identificação” que irá permitir que você faça o *upload* dos seus códigos para o GitHub.

Se você quiser um pouco mais sobre o que é o SSH, que é um protocolo de rede, acesse o link abaixo para ler uma explicação sobre isso diretamente da documentação do GitHub.

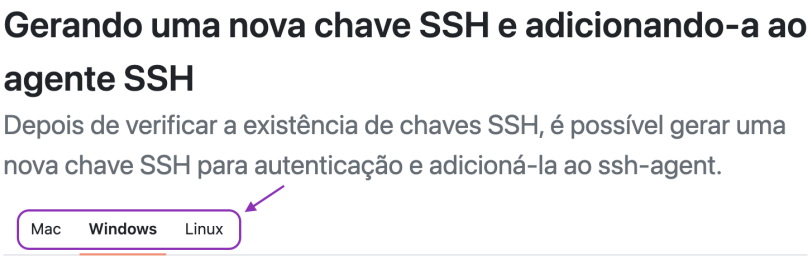
Sobre o SSH - GitHub Docs

Usando o protocolo SSH, você pode se conectar a servidores e serviços remotos e se autenticar neles. Com chaves SSH, você pode se conectar a GitHub sem fornecer seu nome de usuário e personal access token em cada visita. Você também pode usar uma chave SSH para assinar

https://docs.github.com/pt/authentication/connecting-to-github-with-ssh/about-ssh

Agora que você já sabe o que é o SSH, você precisa gerar as suas chaves.

Para isso, eu vou deixar outro link diretamente da documentação do GitHub para que você possa acessar e ver o **passo a passo** de acordo com o seu **sistema operacional**.



Gerando uma nova chave SSH e adicionando-a ao agente SSH - GitHub Docs

Depois de verificar a existência de chaves SSH, é possível gerar uma nova chave SSH para autenticação e adicioná-la ao ssh-agent.

https://docs.github.com/pt/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent?platform=windows

Você pode selecionar o seu sistema operacional nessa parte do site onde deixei a seta roxa.

A importância do README para o seu repositório

Primeiro, o que é README?

📌 README.md é um arquivo com extensão **.md** (Markdown). Contém informações necessárias para **entender o objetivo do projeto**. **README** é uma palavra em ***inglês*** que traduzida fica **LEIAME**.

Você já deve ter baixado algum software que tinha um arquivo **LEIAME.txt** com algumas instruções de como usar.

Podemos considerar o README como um **cartão de visita** do seu projeto no Github ou em outras plataformas de repositórios remotos de código

E o que é Markdown?

📌 O **Markdown** é uma **ferramenta** de conversão de texto em HTML. Você escreve usando texto simples de fácil leitura e fácil escrita e depois é transformado em um HTML válido.

Qual a vantagem de escrever um README?

🎯 Os **recrutadores também observam esse aspecto**. A cultura da empresa desempenha um papel significativo, e muitas delas valorizam muito o uso do GitHub.

Uma vez que a maioria dos recrutadores não possui conhecimento técnico, a simples leitura do seu README em seu repositório pode fornecer a eles uma compreensão geral do que você realizou e uma ideia da complexidade das tecnologias envolvidas.

Este é um **valioso portfólio** e uma maneira eficaz de destacar o seu trabalho. Algumas empresas nem mesmo solicitam currículos; elas preferem avaliar suas habilidades na **prática**. A abordagem mais eficaz é demonstrar suas habilidades por meio de um projeto de teste ou apresentando projetos anteriores.

Como escrever um bom README?

Eu vou deixar alguns links abaixo para te ajudar nesse processo.

readme.so

Use readme.so's markdown editor and templates to easily create a ReadMe for your projects

https://readme.so/pt/editor

Como escrever um bom arquivo README para seu projeto do GitHub

Quando me apresentaram o GitHub, eu não tinha ideia do que era ou do que podia fazer. Cá entre nós, eu criei uma conta porque me disseram que todo desenvolvedor deveria ter uma para poder colocar seu código. Na maior parte do meu tempo de iniciante, eu não fiz nada

https://www.freecodecamp.org/portuguese/news/como-escrever-um-bom-arquivo-readme-para-seu-projeto-do-github

Guia de Comandos Básicos do Git

O Git é uma ferramenta poderosa para o controle de versão de projetos. Para começar a usá-lo efetivamente, é essencial conhecer os comandos básicos. Abaixo estão os comandos mais importantes do Git e suas respectivas explicações:

`git status`

O comando `git status` é seu aliado para verificar o estado atual do seu repositório. Ele fornece informações sobre quais arquivos foram modificados, quais estão pendentes de serem adicionados ou confirmados, e muito mais. Use este comando frequentemente para acompanhar o progresso do seu trabalho.

Exemplo:

```
git status
```

`git log`

O comando `git log` permite visualizar o histórico de commits no seu repositório. Ele exibe informações sobre os commits, como o hash do commit, autor, data e mensagem de confirmação. É útil para entender o que foi feito no projeto e para rastrear alterações.

Exemplo:

```
git log
```

`git add`

Antes de confirmar suas alterações, você precisa "preparar" os arquivos usando `git add`. Este comando coloca as mudanças na área de preparação (*staging area*) para que você possa incluí-las no próximo commit.

Exemplo:

```
git add nome-do-arquivo
```

`git commit`

Após adicionar as mudanças com `git add`, use `git commit` para criar um novo commit com um registro das alterações feitas. Certifique-se de incluir uma mensagem descritiva para que outros colaboradores entendam o motivo da alteração.

Exemplo:

```
git commit -m "Mensagem descritiva da alteração"
```

`git remote`

`git remote` lida com repositórios remotos, que são versões do seu projeto hospedadas em servidores na nuvem ou em outros locais. Você pode listar os repositórios remotos associados ao seu projeto com este comando.

Exemplo:

```
git remote -v
```

`git checkout`

O comando `git checkout` permite alternar entre ramos (branches) no Git. Você pode criar um novo ramo ou mudar para um existente. Isso é útil para trabalhar em recursos isoladamente e evitar conflitos.

Exemplo (criar um novo ramo):

```
git checkout -b nome-do-ramo
```

`git branch`

O comando `git branch` lista todos os ramos disponíveis no repositório. Ele também indica em qual ramo você está atualmente. Isso é útil para ter uma visão geral dos ramos disponíveis.

Exemplo:

```
git branch
```

`git fetch`

`git fetch` é usado para buscar atualizações de um repositório remoto. Ele sincroniza o histórico e as ramificações, mas não aplica as mudanças no seu ramo atual. Use este comando quando desejar atualizar seu repositório com as últimas alterações do repositório remoto.

Exemplo:

```
git fetch origin
```

`git push`

`git push` envia as mudanças confirmadas do seu repositório local para um repositório remoto. Isso é importante para compartilhar seu trabalho com outros colaboradores ou manter um backup atualizado.

Exemplo:

```
git push origin nome-do-ramo
```

`git pull`

`git pull` é usado para buscar e aplicar as mudanças do repositório remoto no seu repositório local. Isso é útil quando você deseja atualizar seu projeto com as últimas alterações feitas por outros colaboradores.

Exemplo:

```
git pull origin nome-do-ramo
```

Lembre-se de que esses são apenas alguns dos comandos essenciais do Git. À medida que você se familiariza com o Git, pode explorar comandos adicionais para realizar tarefas mais avançadas. O Git é uma ferramenta poderosa para o controle de versão, e dominar esses comandos é fundamental para gerenciar projetos de software com eficiência.

Sobre o “origin” do repositório remoto dentro do Git

No contexto do Git, "origin" é o nome padrão dado ao repositório remoto a partir do qual você clonou (copiou) ou com o qual você está colaborando. O "origin" é uma convenção utilizada, mas não é um termo reservado ou uma palavra-chave do Git; você pode dar a um repositório remoto o nome que desejar.

Quando você clona um repositório ou configura um repositório remoto manualmente, geralmente ele é automaticamente nomeado como "origin". Esse nome é apenas uma referência amigável que facilita o trabalho com repositórios remotos, pois é um nome curto e fácil de lembrar.

A principal função do "origin" é permitir que você se comunique com o repositório remoto, facilitando o envio de suas alterações para ele (usando `git push`) e a obtenção das últimas alterações feitas por outros colaboradores (usando `git pull` ou `git fetch`).

Lembre-se de que você pode ter vários repositórios remotos em um único projeto Git, cada um com seu próprio nome (por exemplo, "upstream", "myfork", etc.). O "origin" é simplesmente o nome padrão dado ao primeiro repositório remoto com o qual você está trabalhando, mas você pode adicionar outros repositórios remotos e nomeá-los de acordo com suas necessidades.

Portanto, quando você vê referências a "origin" em comandos Git, está se referindo ao repositório remoto padrão com o qual você está colaborando, mas esse nome pode ser alterado para corresponder à sua configuração específica.

Próximo módulo ➡️SOON

Wooow, parabéns por ter chegado até aqui!

O aprendizado do Git é algo que depende muito da repetição e prática. Por isso, sempre que possível, esteja praticando e revisando os comandos.

Caso tenha ficado alguma dúvida, aproveite para revisar as aulas e não esqueça de responder aos exercícios deste módulo.

Agora, no próximo módulo vamos falar sobre como **“Fundamentos da programação web”**, e assim vamos aprender sobre toda a base de protocolos e arquitetura por de baixo dos panos da internet.

Com esse conhecimento, você será capaz de entender toda a sequência de aprendizado que vamos ter nos módulos seguintes e colocar a mão na massa. Até a próxima!