UNIVERSIDADE ESTADUAL DO CENTRO-OESTE
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Paulo Silvério, Yorranan Costa

Git e *GitHub* – controle de versão para iniciantes

1. *Git* e *Github*

O *Git* e o *GitHub* são duas ferramentas relacionadas, mas distintas, que desempenham papéis importantes no desenvolvimento de *software* colaborativo e no controle de versões.

1.1. Git

O *Git* é um sistema de controle de versão distribuído, projetado para lidar com projetos de qualquer tamanho, desde pequenos até os mais complexos. Ele permite que você controle e acompanhe as alterações feitas em arquivos e diretórios ao longo do tempo, facilitando o trabalho em equipe, a colaboração e o gerenciamento eficiente do código-fonte. Com o *Git*, você pode rastrear as alterações feitas em seu projeto, criar diferentes versões do seu código (ramificações) e mesclá-las conforme necessário.

O *Git* é amplamente utilizado devido às suas características poderosas, como velocidade, eficiência e suporte a desenvolvimento não linear. Ele funciona localmente no seu computador e permite que você trabalhe offline, realizando operações como confirmar mudanças, ramificar e mesclar diferentes trechos de códigos. Em seguida, você pode sincronizar suas alterações com repositórios remotos, como o *GitHub*.

1.2. GitHub

O *GitHub*, por sua vez, é uma plataforma de hospedagem de código-fonte baseada em nuvem que permite que desenvolvedores e equipes compartilhem, colaborem e trabalhem em projetos de software usando o *Git*. É um serviço que fornece um repositório centralizado para seus projetos *Git,* onde você pode armazenar, gerenciar e compartilhar seu código com outros desenvolvedores. O *GitHub* oferece recursos adicionais, como controle de acesso, gerenciamento de problemas (*issues*), integração contínua e ferramentas de colaboração, como *pull requests*, que facilitam o trabalho em equipe.

1.3. Git + Github: Uso em conjunto

Ao usar o *Git* com o *GitHub*, você pode fazer o seguinte:

- Controle de versão: O *Git* é especialmente conhecido por seu excelente controle de versão. Ele permite que você rastreie todas as alterações feitas nos arquivos ao longo do tempo, facilitando a visualização de revisões anteriores, correção de *bugs* e recuperação de versões específicas do código.

- Branching e Merging: Com o Git, você pode criar ramificações (*branches*) para desenvolver novos recursos ou corrigir *bugs* sem afetar o código principal. Em seguida, você pode mesclar (*merge*) suas alterações de volta ao *branch* principal por meio de *pull requests* ou *merge requests*.

- Colaboração: O GitHub é uma plataforma de colaboração que permite que várias pessoas trabalhem juntas em um projeto. Os membros da equipe podem contribuir com código, revisar alterações uns dos outros e fornecer *feedback* por meio de *pull requests*.

- *Issues* e *Projects*: O GitHub possui um sistema de gerenciamento de problemas (*issues*) integrado, que permite que você rastreie tarefas, bugs ou solicitações de recursos. Além disso, você pode usar os projetos (*projects*) para organizar e priorizar tarefas em um quadro Kanban.

- Integração Contínua: O GitHub permite integrar seu repositório com ferramentas de integração contínua (CI), como Travis CI ou GitHub Actions. Isso significa que sempre que você faz um *commit*, seu código é automaticamente testado e compilado para garantir a integridade do projeto.

- Versionamento Semântico: O Git, juntamente com o GitHub, facilita o uso de versionamento semântico, permitindo que você atribua *tags* específicas a versões importantes do seu projeto, como lançamentos (*releases*).

- Wiki e Documentação: O GitHub permite criar um documento em formato wiki para documentar seu projeto e fornecer informações adicionais para os colaboradores.

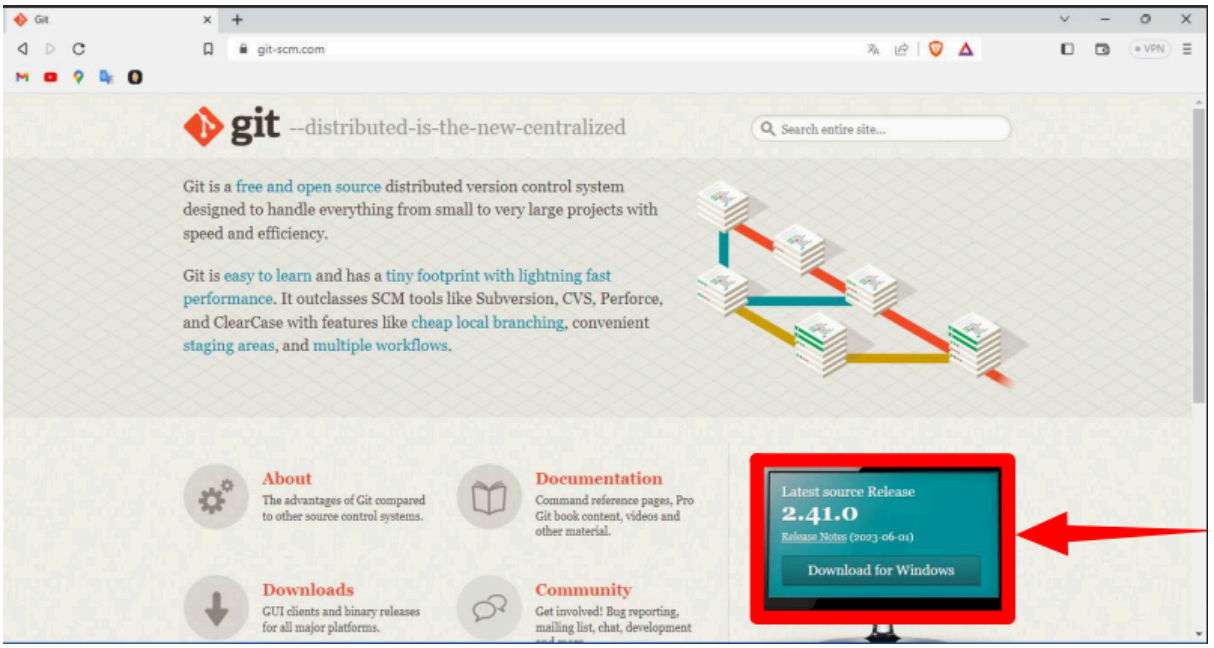
- Gists: O GitHub também permite criar *gists*, que são pequenos trechos de código ou notas que podem ser compartilhados e incorporados em outros sites.

- *Fork* e *Pull Requests*: Você pode criar uma cópia independente de um projeto na sua conta. Após fazer modificações no *fork*, você pode enviar um *pull request* para o repositório original, solicitando que suas alterações sejam mescladas ao projeto principal.

- Segurança: O *GitHub* possui recursos de segurança que permitem analisar e identificar vulnerabilidades em seu código, ajudando a manter seu projeto mais seguro.

2. Instalando *Git* no *Windows*

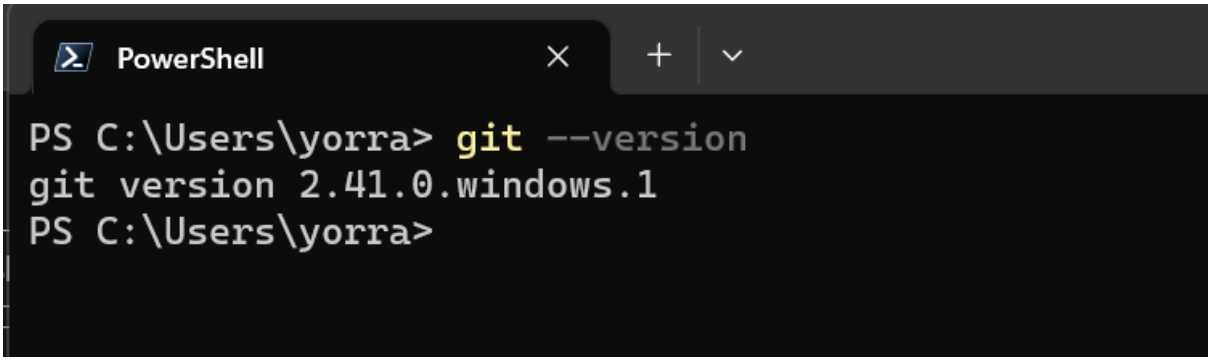
1. Acesse o site oficial do *Git* em https://git-scm.com/.



2. Procure o *link* de *download* para o seu sistema operacional e clique nele para iniciar o *download*.

3. Depois que o *download* for concluído, execute o instalador do *Git* e siga as instruções na tela.
4. Durante a instalação, você pode aceitar as configurações padrão, a menos que tenha um motivo específico para alterá-las.
5. Após a conclusão da instalação, abra o terminal ou *prompt* de comando para verificar se o *Git* foi instalado corretamente. Digite o seguinte comando e pressione Enter:

```
git –version
```



3. Instalando *Git* no *Linux*

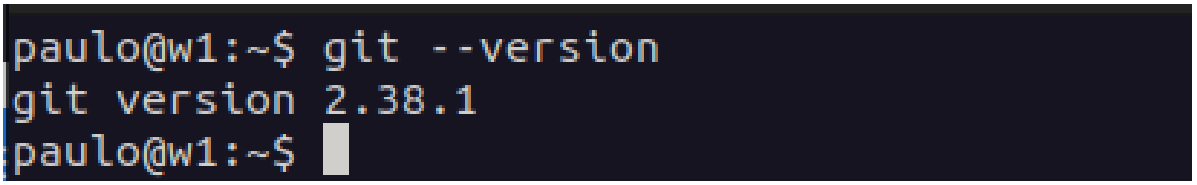
Para instalar o *Git* em diversas distribuições *Linux*, você pode usar o gerenciador de pacotes específico de cada uma. Aqui estão os comandos para algumas das distribuições mais populares.

Abra o terminal no seu sistema *Linux*. Você pode fazer isso pressionando as teclas "Ctrl + Alt + T" simultaneamente.

- *Ubuntu* e *Debian:*
  `sudo apt install git`
- *Fedora*:
  `sudo dnf install git`
- *CentOS/RHEL (Red Hat Enterprise Linux)/Oracle Linux*:
  `sudo yum install git`
- *openSUSE/SUSE Linux Enterprise*:
  `sudo zypper install git`
- *Arch Linux/Manjaro*:
  `sudo pacman -Syu git`
- *Gentoo*:
  `sudo emerge --ask --verbose dev-vcs/git`

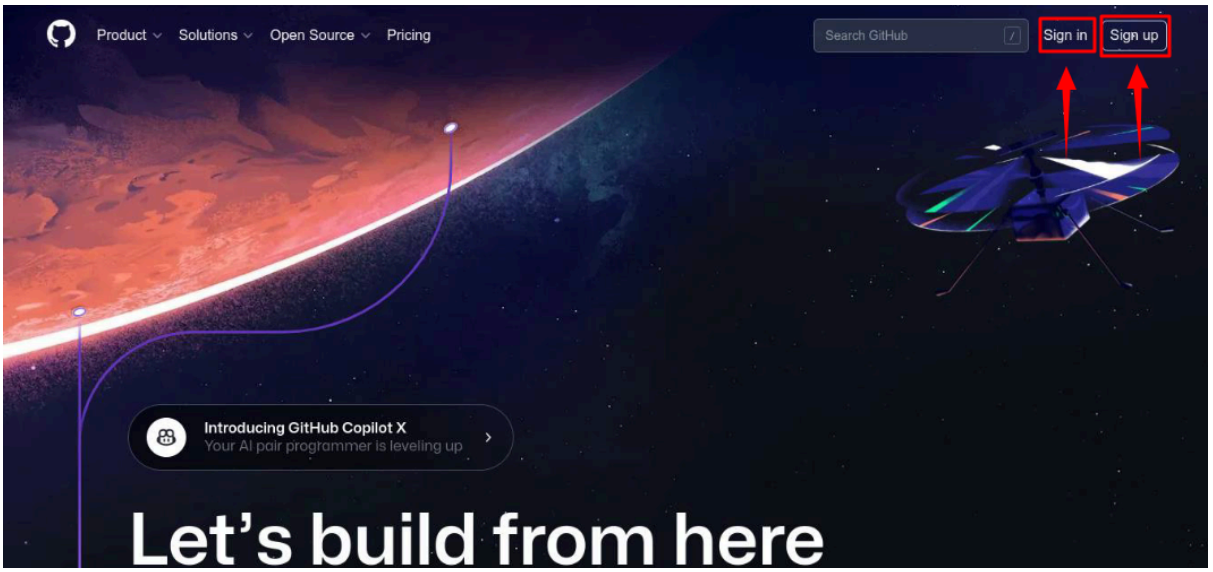
Após a conclusão da instalação, verifique se o Git foi instalado corretamente. Digite o seguinte comando e pressione *Enter*:

`git –version`



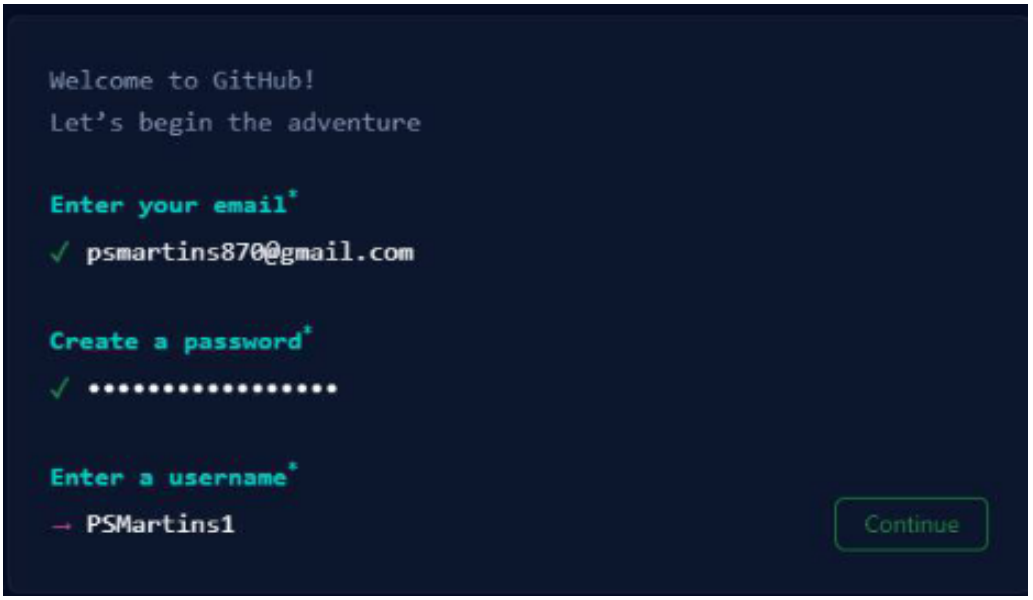
4. Criando Conta no *GitHub*

1. Abra o seu navegador e vá para o site do *GitHub*: https://github.com
2. Na página inicial do *GitHub*, você verá um campo de inscrição no canto superior direito.



3. Preencha as informações necessárias:

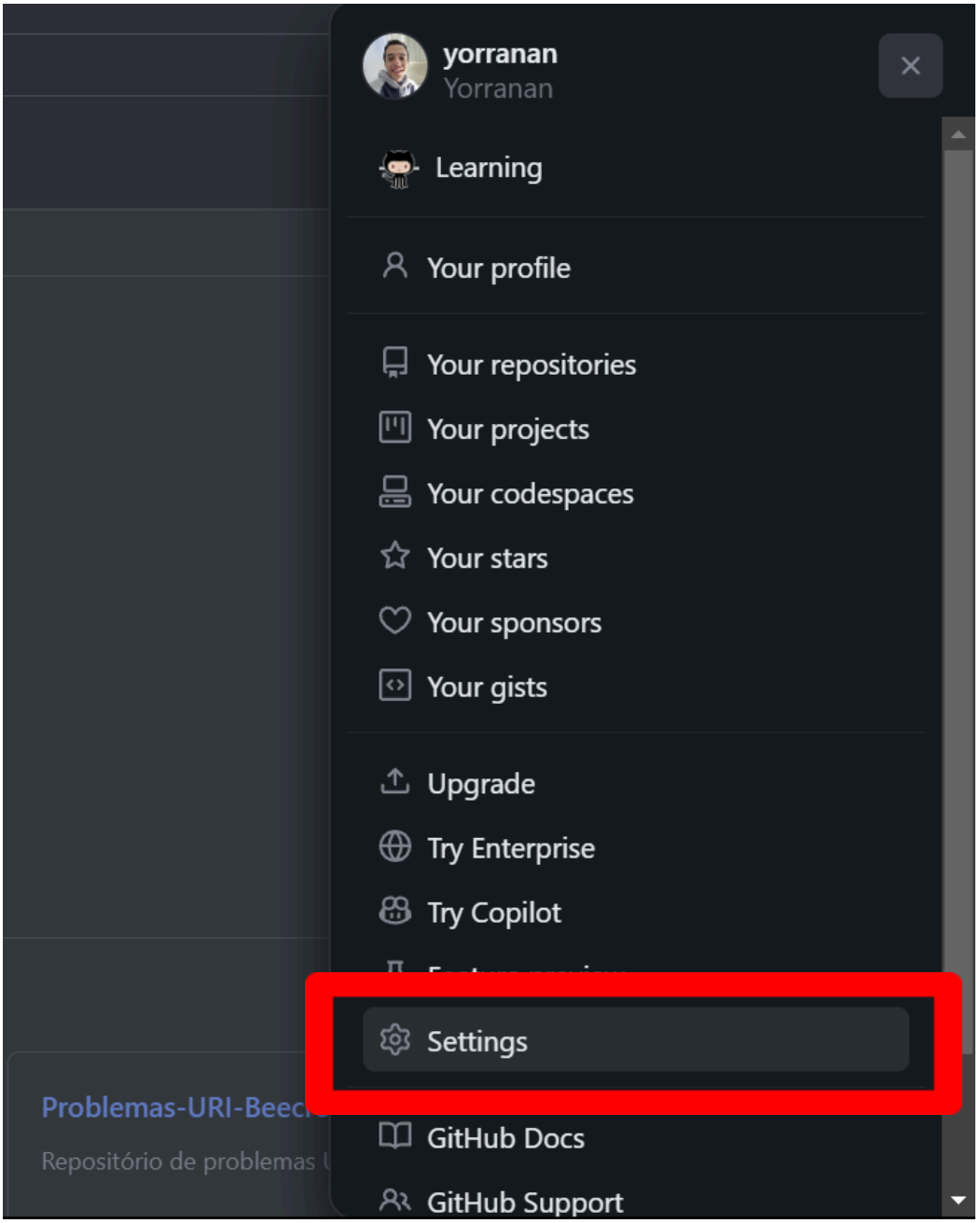
- Escolha um nome de usuário: Digite um nome de usuário único que você deseja usar no *GitHub*.
- Digite seu endereço de *e-mail*: Insira um endereço de *e-mail* válido que você tenha acesso.
- Crie uma senha: Escolha uma senha segura para proteger sua conta.



5. Configurando autenticação via *token*.

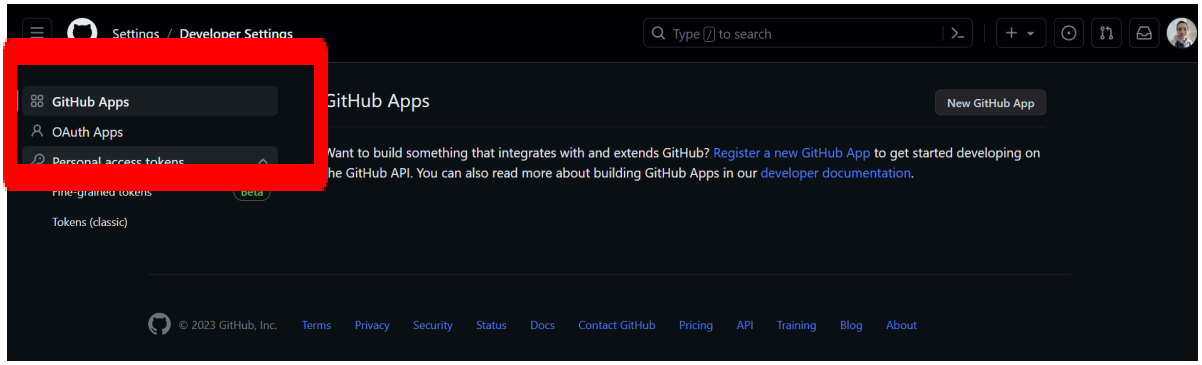
Desde 2021 o *GitHub* tornou obrigatório o uso do *token* para autenticar operações. Para usar o *token* é necessário estar com seu *e-mail* verificado na plataforma. Após isso siga os seguintes passos como cita a documentação oficial:

1) No canto superior direito onde clique na foto do seu perfil e procure **configurações**.

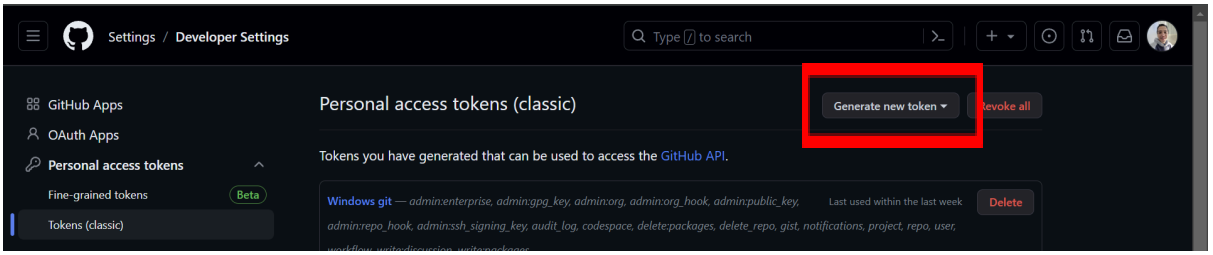


2) Após isso vá até o final da página de configurações e na barra lateral esquerda, clique em **Configurações do desenvolvedor**.

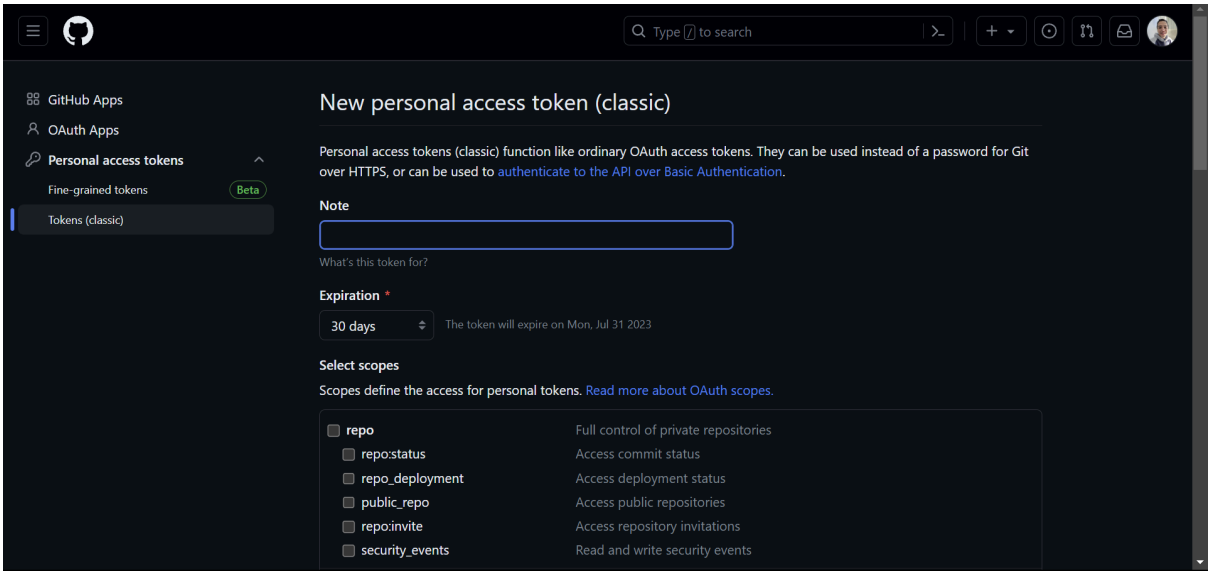
3) Na barra lateral esquerda, em ***Personal access tokens***, clique em ***Tokens* clássicos.**



4) Clique em **Gerar novo *Token***

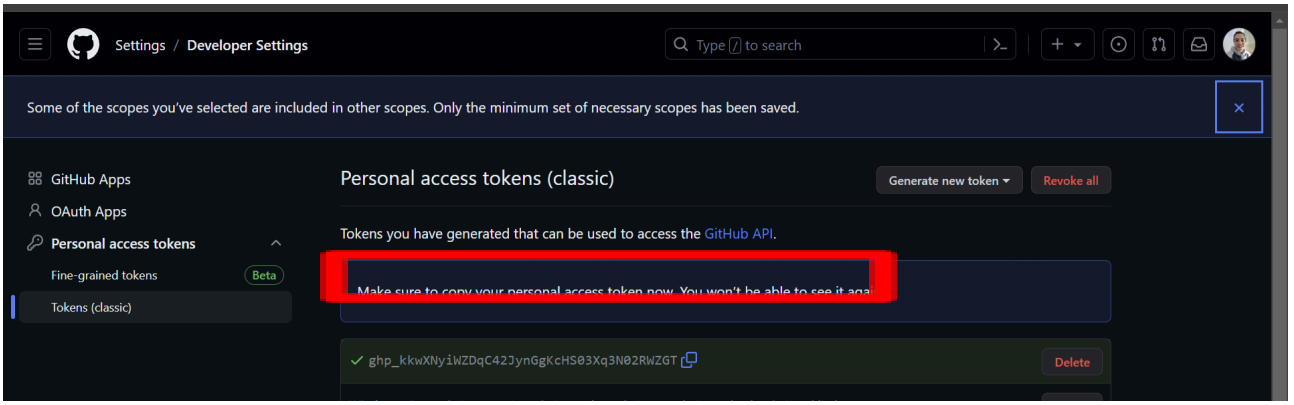


5) Aparecerá uma tela com diversas opções.



6) Em Nota do *token*, insira uma nota para ele.
7) Em **Expiração**, selecione uma validação para o *token*.
8) Para ter acesso completo as funcionalidades marque todas as opções disponíveis, contudo se deseja personalizar adequadamente confira a documentação que pode ser acessada através de: https://docs.github.com/pt/enterprise-cloud@latest/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens

9) Por fim, copie o *token* gerado, que será usado para autenticar as operações.



5.1. Configurando usuário no computador

Através do *PowerShell* ou terminal é necessário realizar as configurações de uso, para isso é necessário usar os seguintes comandos:

```
git config --global user.name "Fulano de Tal"
git config --global user.email fulanodetal@exemplo.br
```

Esses comandos inserem um nome de usuário e *email* para identificar quem modifica os códigos de maneira global.

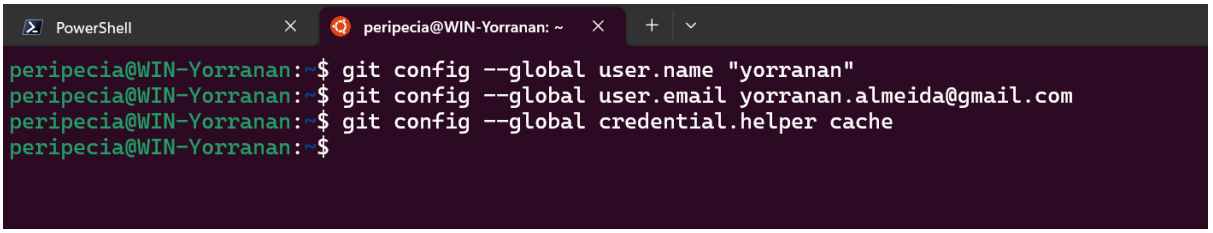
É importante salientar que isso é a configuração do *Git* não do *GitHub*, outro ponto importante é que esse comando deve ser executado toda vez que o usuário for usar o git, uma vez que se o usuário não estiver logado não poderá fazer alterações no repositório, para fazer *log out* usar o comando:

```
git config --global --unset-all
```

Para evitar a necessidade de informar o *token* toda vez que fizermos operações usando o *GitHub*, você pode salvar no seu computador essa informação usando o seguinte comando:

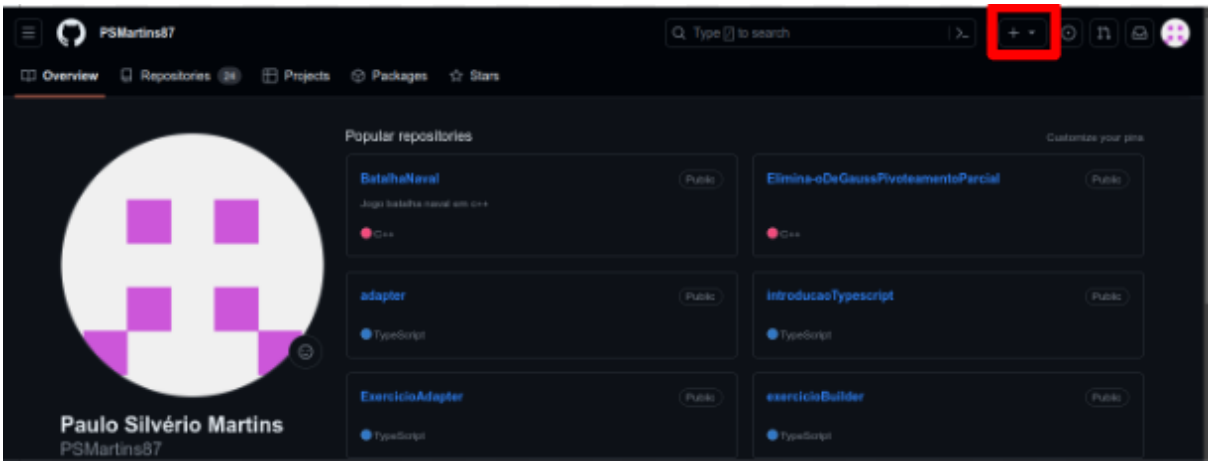
```
git config --global credential.helper cache
```

Após isso ele salvará a próxima credencial inserida pelo usuário. Exemplo:

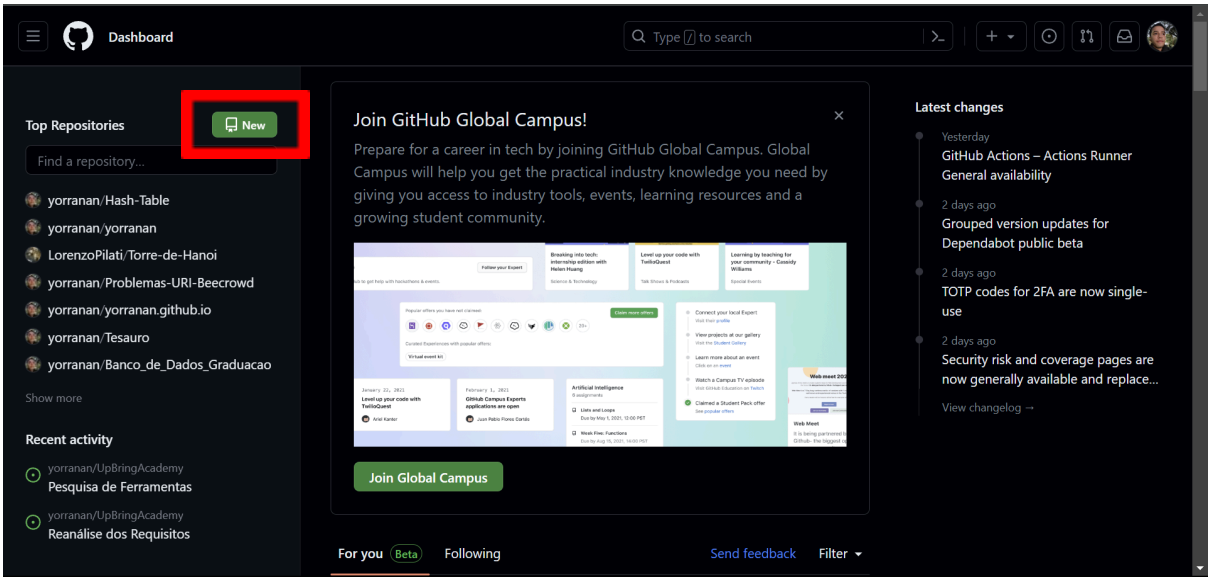


6. Criando Repositório no GitHub

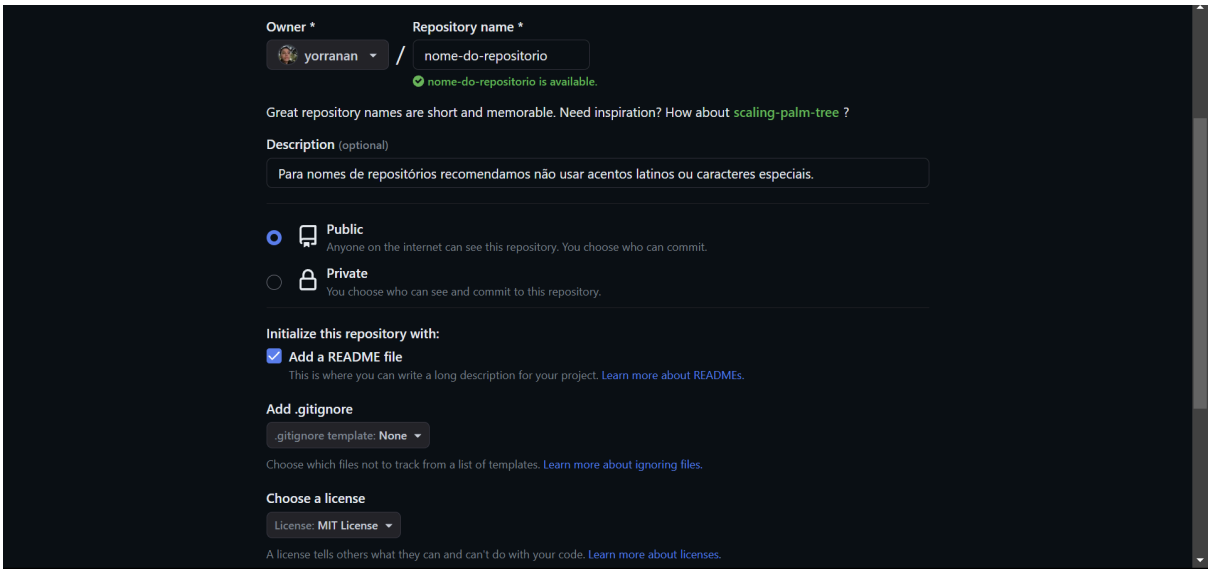
- No canto superior direito da página (de seu perfil), clique no botão "+", ao lado da sua foto de perfil, e selecione "Novo repositório" no menu suspenso.



- Se você estiver na página inicial procure o botão verde escrito “novo” ou “new” localizado na barra lateral esquerda.



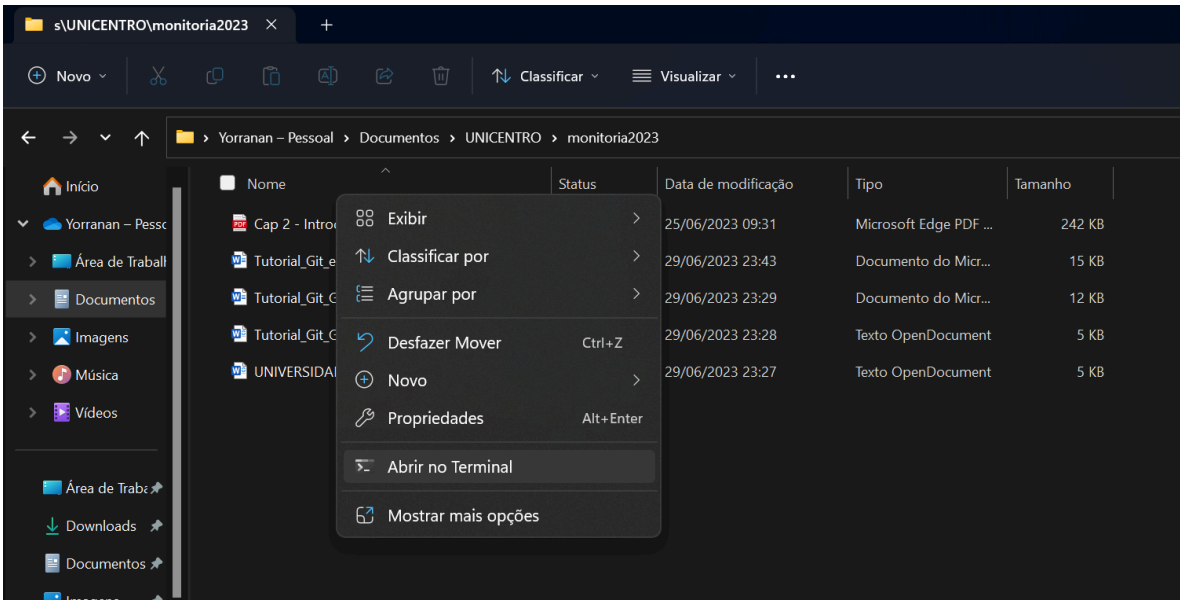
- Na página "Criar um novo repositório", preencha as informações básicas do seu repositório.
- No campo "Nome do repositório", digite um nome descritivo para o seu projeto.
- Escreva uma breve descrição do repositório no campo "Descrição" (opcional).
- Escolha se o repositório será público (visível para todos) ou privado (visível apenas para você ou pessoas específicas, se você tiver uma conta paga).
- Se desejar, selecione a opção "Inicializar este repositório com um arquivo *README*" para criar automaticamente um arquivo *README.md* inicial para o seu repositório. O arquivo *README.md* é usado para fornecer informações sobre o projeto.
- Você também pode adicionar um arquivo de licença, um arquivo *.gitignore* específico e outros arquivos adicionais, se desejar.
- Após preencher as informações, clique no botão "Criar repositório". Seu repositório será criado no *GitHub*.



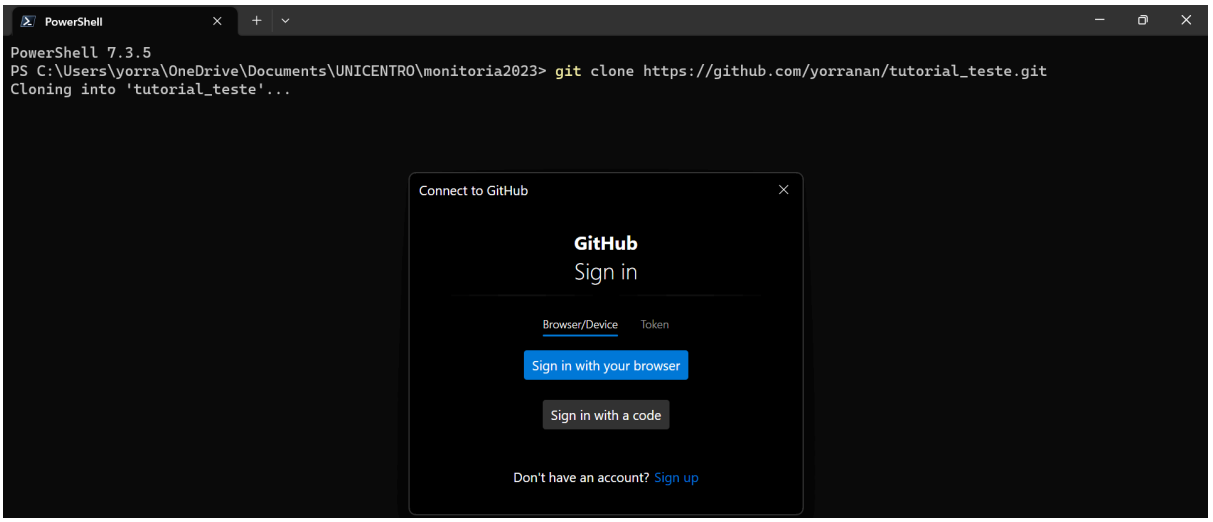
Agora, você pode adicionar arquivos ao seu repositório, cloná-lo para o seu computador local, fazer commits e push das suas alterações, e muito mais. O *GitHub* oferece instruções detalhadas para começar a trabalhar com o seu novo repositório, incluindo comandos *Git* que você pode usar.

6.1. Usando o novo repositório criado

No *Windows* 11 pode-se abrir a pasta onde se deseja inserir o novo repositório, clicando com o botão direito selecione a opção de abrir no terminal. Em grande parte das distribuições existem opções similares usando a interface gráfica.

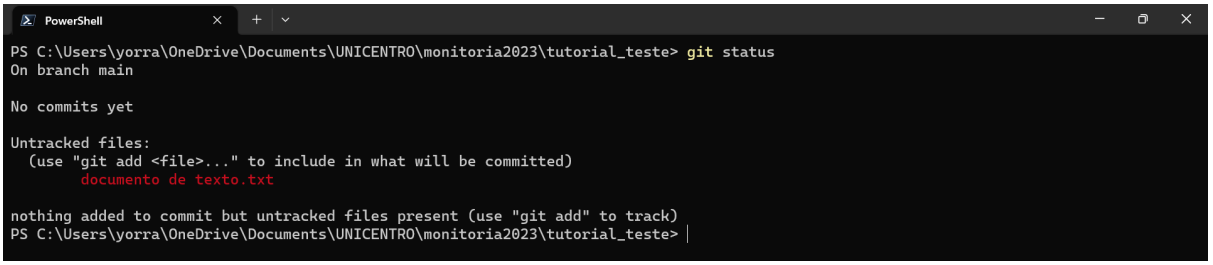


No Windows poderá aparecer uma tela de autenticação similar a essa após usar `git clone`.



Se essa tela não aparecer por quaisquer motivos, o terminal perguntará seu usuário e você deve inserir o mesmo e-mail cadastrado no GitHub. Após isso ele deve pedir a senha ou autenticação, nesse caso cole o *token* que você copiou após criá-lo no capítulo 5.

Após isso, qualquer modificação que você realizar na pasta clonada permanecerá localmente até que você envie um *commit*. Dentro da pasta do repositório que você clonou, a partir do terminal deve-se verificar as modificações usando o comando *git status*.



Pode-se adicionar os arquivos individualmente ou todos através do *git add **.

```
PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste> git add *
PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste> git stat
git: 'stat' is not a git command. See 'git --help'.

The most similar commands are
        status
        stage
        stash
PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste> git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   documento de texto.txt

PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste>
```

Feito isso, os arquivos em verde estão prontos para serem "empacotados" usando o comando *commit*.

```
PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste> git commit -m "A mensagem"
[main (root-commit) dfb22bc] A mensagem
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 documento de texto.txt
PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste>
```

Feito as operações acima podemos finalmente realizar "empurrar" nosso *commit* para o repositório no *GitHub* usando *push*.

```
PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste> git commit -m "A mensagem"
[main (root-commit) dfb22bc] A mensagem
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 documento de texto.txt
PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste> git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 224 bytes | 224.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/yorranan/tutorial_teste.git
 * [new branch]      main -> main
PS C:\Users\yorra\OneDrive\Documents\UNICENTRO\monitoria2023\tutorial_teste>
```

Dependendo de como o *git* está instalado em sua máquina, ou se você está usando *Linux*, é nesse momento que será feita a autenticação com a conta do *GitHub*, então preste muita atenção.

7. Comandos do *git*

Os comandos do *Git* são um conjunto de instruções utilizadas para interagir com o sistema de controle de versão distribuído *Git*, todos os comandos começam com a palavra *git*, usando o comando `git –help` no terminal é possível ver mais comandos e suas finalidades , segue a abaixo os mais usados:

7.1. *git init*

Sintaxe: `git init`
Descrição: Inicializa um novo repositório Git no diretório atual.

Esse comando é usado para inicializar um novo repositório *Git* em um diretório local. Ele cria uma estrutura .git, que é responsável por armazenar o histórico de alterações e os metadados do projeto.

7.2. git clone

Sintaxe: `git clone <URL do repositório>`
Descrição: Clona um repositório existente do Git para o seu computador local. Isso cria uma cópia local do repositório completo, incluindo todo o histórico de alterações.
Exemplo: `git clone https://github.com/usuario/repositorio.git`

7.3. git status

Sintaxe: `git status`
Descrição: Mostra o status atual do seu repositório *Git*, exibindo as alterações pendentes, os arquivos adicionados, entre outros detalhes.

7.4. git add

Sintaxe: `git add <arquivo>`
Descrição: Adiciona arquivos ao índice do *Git* para serem incluídos no próximo *commit*. O índice funciona como uma área de preparação para as alterações que você deseja commitar.
Exemplo: `git add arquivo.txt`

7.5. git rm

Sintaxe: `git rm <arquivo>`
Descrição: É usado para remover arquivos do diretório de trabalho e do índice do *Git*, ele pode ser usado para excluir arquivos que não são mais necessários no repositório.
Exemplo: `git rm arquivo.txt`

7.6. git commit

Sintaxe: `git commit -m "Mensagem do commit"`
Descrição: Grava um *snapshot* das alterações feitas nos arquivos adicionados ao índice. É uma forma de confirmar as alterações e torná-las parte do histórico do projeto. Você pode adicionar uma mensagem descritiva ao *commit* para registrar o propósito das alterações.
Exemplo: `git commit -m "Adiciona novas funcionalidades"`

7.7. git push

Sintaxe: `git push <repositório-remoto> <branch>`
Descrição: Envia as alterações locais para um repositório remoto. É usado para compartilhar as alterações feitas no seu projeto com outros colaboradores ou com o repositório central.
Exemplo: `git push origin main`

7.8. git restore

8. Bibliografia básica com alguns recursos úteis para quem deseja aprender mais sobre o Git e o GitHub

- Pro Git (livro): Escrito por Scott Chacon e Ben Straub, o livro "Pro Git" é uma referência abrangente e detalhada sobre o Git. Ele aborda desde conceitos básicos até tópicos avançados, oferecendo uma visão completa do sistema de controle de versão distribuído. O livro está disponível gratuitamente online em vários idiomas, incluindo inglês, espanhol, português (parcial) e outros. [Link: https://git-scm.com/book]

- Documentação oficial do Git: A documentação oficial do Git é uma excelente fonte de informações. Ela fornece uma visão abrangente dos comandos, opções e recursos do Git, além de exemplos e tutoriais passo a passo. A documentação está disponível em vários idiomas, incluindo inglês, espanhol, português e outros. [Link: https://git-scm.com/doc]

- GitHub Guides: O GitHub possui uma seção de guias (Guides) que oferece tutoriais e informações úteis sobre como usar o Git e o GitHub em diferentes contextos. Os guias abrangem desde

conceitos básicos até fluxos de trabalho avançados e integração contínua. [Link: https://guides.github.com/]

- *Git Cheat Sheet*: Um "*cheat sheet*" é um resumo rápido e prático das principais informações sobre um tópico. O *Git Cheat Sheet* é um recurso útil para ter à mão, fornecendo uma visão geral dos comandos mais comuns do Git e suas sintaxes. Existem várias versões disponíveis online em diferentes idiomas. [Exemplo: https://education.github.com/git-cheat-sheet-education.pdf]

9. Apêndices

9.1. Comandos de terminal úteis para desenvolvedores

9.1.1. Comandos do Terminal para Windows:

- `cd`: Permite navegar entre diretórios. `Use cd ..` para voltar um diretório e `cd nome_do_diretório` para entrar em um diretório específico, ou cd sem argumentos ele exibirá o diretório em que você está atualmente localizado.

- `dir`: Lista os arquivos e diretórios no diretório atual.

- `mkdir`: Cria um novo diretório. Por exemplo, `mkdir nome_do_diretório` cria um diretório com o nome especificado.

- `del`: Exclui arquivos. Use `del nome_do_arquivo` para excluir um arquivo específico.

- `cls`: Limpa a tela do prompt de comando.

- `type`: Exibe o conteúdo de um arquivo de texto. Por exemplo, `type nome_do_arquivo.txt` mostra o conteúdo do arquivo de texto.

- `ren`: Renomeia arquivos. Use `ren nome_do_arquivo novo_nome` para renomear um arquivo.

9.1.2. Comandos do Terminal para Linux:

- `cd`: Também usado para navegar entre diretórios no Linux, da mesma forma que no Windows.

- `ls`: Lista os arquivos e diretórios no diretório atual.

- `mkdir`: Similar ao comando do Windows, cria um novo diretório.

- `rm`: Exclui arquivos ou diretórios. Use `rm nome_do_arquivo` para excluir um arquivo e `rm -r nome_do_diretório` para excluir um diretório.

- `clear`: Limpa a tela do terminal.

- `cat`: Exibe o conteúdo de um arquivo de texto. Por exemplo, `cat nome_do_arquivo.txt` mostra o conteúdo do arquivo de texto.

- `mv`: Move ou renomeia arquivos. Use `mv nome_do_arquivo novo_nome` para renomear um arquivo e `mv nome_do_arquivo caminho_destino` para mover um arquivo para um diretório específico.

Fontes bibliográficas

CHACON, Scott; STRAUB, Ben. Pro Git. Apress, Berkeley, CA, 2014. Disponível em: https://git-scm.com/book/en/v2. Acesso em: 5 jul. 2023.

GITHUB, Inc. Understanding the GitHub flow. GitHub Guides, San Francisco, v. 3, n. 1, p. 1-7, jan. 2019. Disponível em: https://guides.github.com/introduction/flow/. Acesso em: 1 jul. 2023.