Comandos Iniciais

Criando sua conta no GitHub

Para realizar esse curso, você precisa ter uma conta no GitHub. Caso ainda não tenha, acesse o site do GitHub e clique no botão *Sign up*.

Isso abrirá um formulário a ser preenchido com suas informações. Também haverá um processo de verificação, para garantir que você não é um robô. :)

Criada a conta, o GitHub **enviará um código para o seu e-mail**. Verifique seu e-mail, inclusive as abas de “Promoções”, “Social” e “Spam”. Após encontrá-lo, informe o código enviado.

Assim, sua conta no GitHub estará criada e pronta para uso.

Configurando seu usuário do GitHub localmente

Caso tenha dúvida se tem a confiugação execute o comando:

```
git config --list
```

Por fim, você precisará configurar seu usuário do GitHub em seu computador. Para isso, abra seu Terminal e execute os seguintes comandos:

```
git config --global user.name "SEU NOME"
git config --global user.email EMAIL@exemplo.br
```

Lembre-se de substituir “SEU NOME” pelo seu nome e “EMAIL@exemplo.br” pelo e-mail que você usou na sua conta do GitHub.

Inicializando um repositório Git

Git e GitHub

O GitHub usa o sistema de versionamento Git, que permite rastrear as alterações feitas ao longo do projeto, além de permitir trabalho colaborativo.

É importante diferenciar *Git e GitHub.*

Git é um sistema de versionamento, enquanto GitHub é uma plataforma que usa esse sistema para armazenar nosso código.

Vamos subir o código do nosso computador para o GitHub.

Criando um repositório

Precisamos **criar um repositório**. Para isso, clicamos no ícone no canto superior direito, selecionamos "*Your Repositories*" ("Seus repositórios") e clicamos no botão verde "*New*" ("Novo") do lado direito.

Isso abrirá opções para a criação do novo repositório em uma página intitulada "*Create a new repository*" ("Crie um novo repositório").

O campo *"Owner"* ("Proprietário") indica quem é a dona do repositório, e o campo *"Repository Name"* é o nome do repositório. Chamaremos o repositório de ***"iec-git"***.

As demais configurações para o repositório podem ser mantidas como estão. Clicamos no botão verde *"Create Repository"* ("Criar repositório") no canto inferior direito para criar o **repositório remoto**, sendo o que está na *web*.

O *GitHub* oferece algumas orientações sobre como podemos enviar o código de nossa máquina para esse repositório remoto.

Iniciando um repositório Git no projeto local

Precisamos iniciar um repositório *Git* em nosso projeto local. Para isso, executamos o comando `git init`.

Definindo *branch*

Em seguida, definimos um *branch* com o comando `git branch -M main`.

Conectando o repositório local com o do GitHub

Agora, vamos conectar o repositório que criamos no GitHub com o nosso repositório local em nosso computador. Voltamos ao navegador, para o repositório que criamos e copiamos a URL fornecida no campo "Quick Setup" ("Configuração rápida"). EX

```
https://github.com/negoNegoso/iec-git.git
```

De volta ao terminal do *VSCode*, executamos o comando `git remote add`.

Precisamos dar um nome a esse repositório remoto, geralmente chamamos de `origin`. Informamos a *URL* do repositório.

```
git remote add origin https://github.com/negoNegoso/iec-git.git
```

Rodamos o comando `cls` para limpar a tela e depois `git remote`.

O terminal indicará que temos um repositório remoto chamado *origin*.

Adicionando, Registrando e Enviando Alterações no *GitHub*

Fechamos o explorador, abrimos o terminal e executamos alguns comandos. Ao rodarmos o comando `git status`, poderemos visualizar os arquivos que foram modificados ou adicionados em nosso projeto e ainda **não foram incluídos no GitHub** - eles serão destacados em **vermelho**.

Adicionando arquivos

Em seguida, utilizaremos o comando `git add` para especificar quais arquivos desejamos adicionar.

Podemos simplificar o processo de adicionar arquivos ao nosso projeto. Em vez de digitar manualmente o nome de cada arquivo (por exemplo, `git add README.md` e teclar "*Enter*"), podemos usar um atalho chamado `git add .`.

Esse comando adiciona **todos os arquivos de uma vez**. Ao rodarmos novamente o comando `git status`, veremos a lista de todos os arquivos que foram adicionados.

Criando um *commit*

Agora, criaremos um `commit` para registrar as mudanças realizadas. Utilizamos o comando `git commit`, incluindo a opção "`-m`" para adicionar uma mensagem sobre as alterações feitas no projeto.

```
git commit -m
```

Após o espaço, colocamos aspas duplas e escrevemos uma breve descrição das modificações. Como mensagem, optamos por "`initial commit`". Fechamos as aspas e pressionamos "Enter".

```
git commit -m "initial commit"
```

GitHub - iuricode/padroes-de-commits: Padrões de commits

Padrões de commits. Contribute to iuricode/padroes-de-commits development by creating an account on GitHub.

O comando de `commit` **registra a alteração** realizada. Podemos limpar a tela de novo com `cls`.

Podemos executar o comando `git log` para visualizar o registro das alterações feitas em nosso projeto.

```
git log
```

Enviando para o *GitHub*

O último passo é utilizar o comando `git push`, especificando o destino como o nosso `origin main`.

```
git push origin main
```

Teclamos "Enter". Aparentemente ocorreu tudo bem.

Nos arquivos, selecionamos o `README.md` e efetuamos a modificação diretamente no *GitHub*. Acima do arquivo, encontramos várias opções e no canto direito, identificamos um botão com um ícone de lápis para editar o arquivo (ao colocarmos o *mouse* por cima obtemos a mensagem "*Edit this file*").

```
:tada:  Boas-vindas ao Github
```

Subimos um pouco na tela e temos esse botão verde na lateral direita de "*Commit Changes*" ("Confirmar alterações"). Clicamos nesse botão para criar um *commit* diretamente pelo *GitHub*.

Será exibida uma janela intitulada "*Commit changes*" com os campos "*Commit message*" e "*Extended description*" ("Descrição estendida"). Na parte inferior direita, temos um botão "*Cancel*" e outro denominado "*Commit changes*".

Vamos deixar a mensagem que está aqui (*Update README.md*), mas se preferir, você pode alterar essa mensagem e clicamos no botão verde de "*Commit Changes*".

Enquanto no *GitHub* temos dois *commits*, no terminal temos apenas um. É necessário **sincronizar** essas alterações em nossa máquina, utilizando o comando `git pull origin main`.

```
git pull origin main
```

Esse é o fluxo de trabalho que geralmente temos quando usamos o *GitHub*: temos um projeto no nosso computador, fazemos alterações no nosso código do nosso projeto, criamos um *commit*, enviamos essas alterações para o nosso *GitHub*, para o nosso repositório remoto com o comando `git push`.

Nessa aula, você aprendeu:

- A criar um repositório git local com os comandos `git init` e `git branch -M main` e o repositório remoto para poder compartilhar o seu código com outras pessoas;
- A conectar um repositório local com uma remoto através do comando `git remote add origin URL`;
- Que o Git e o GitHub são diferentes. Enquanto o git é a ferramenta responsável pelo controle do versionamento de códigos, o GitHub é a plataforma responsável pelos repositórios;
- Que criamos um commit para registrar as alterações realizadas no projeto com o comando `git commit -m “Mensagem que descreve as mudanças”`;
- A enviar e trazer alterações feitas para os repositórios local e remoto por meio dos comandos `git push origin main` e `git pull origin main`;
- A cuidar do fluxo de trabalho em projetos colaborativos usando o comandos git, como `git status`, `git add`, `git commit`, `git push` e `git pull`.