

GIT & GITHUB, ATRASSO & INTERVALO, OBJETO & JSON



DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

- **Versionamento**
 - Revisão & Termos Git
 - GitHub & GitHub Desktop
- **Intervalos de tempo**
 - setTimeout & setInterval
- **Objetos**
- **JSON**

- **Versionamento**
 - O que é?
 - Para que serve?
 - Que problemas veio resolver?

- **repositório** (repository)
 - Instância principal da base de código de algum software
- **remote/origin** (remoto/origem)
 - Instância centralizadora do repositório que está no servidor
- **local**
 - Instância do repositório que está em nossa máquina local
- **branch** (ramo)
 - Ramificação nomeada do repositório a partir de um commit

- **commit** (compromisso/entrega)
 - Pacote de mudanças a serem registradas/integradas
- **push** (empurrar)
 - Ação de envio das alterações locais para o remoto
- **pull** (puxar)
 - Ação de "baixar" alterações do remoto para o local
- **fetch** (buscar)
 - Ação de buscar branches/tags/dados atualizadas do projeto

- **Pull request** (Pedido de "pull")
 - Ação de solicitar revisão das alterações de uma branch para que possam ser mescladas com uma outra branch
- **merge** (mesclar/fundir)
 - Ação de mesclar/fundir o conteúdo de uma branch com outra para que os commits sejam transferidos à outra branch
- **rebase** (rebasear)
 - Ação de redefinir o commit de origem da branch para atualizar uma branch com os últimos commits de sua branch de origem

- O GitHub possui um cliente com interface gráfica, para que a gente possa utilizar o Git sem precisar da linha de comando.
- Faça o download do "**GitHub Desktop**" através da URL: <https://desktop.github.com>
- Agora vamos abrir o GitHub Desktop e fazer login com a nossa conta GitHub.
- Em alguns casos pode ser necessário baixar o Git via <https://git-scm.com/downloads>

VERSIONAMENTO | GitHub

A TINY CHEATSHEET ON

GIT



git is the most popular version control system. Being proficient in git is a must for every developer.

git init

Create an empty repository in the current directory

git add [file]

Add the **[file]**(s) to the *staging area*

git commit

Create a snapshot of all the files in the *staging area*

git push [name] [b]

Push changes to a remote repository called **[name]** to branch **[b]**

git pull [name] [b]

Pull any changes from a remote repository called **[name]** from branch **[b]**

git branch

git branch

List all of the branches in your repo. Add a **[branch]** argument to create a new branch called **[branch]**.

git checkout -b [branch]

Switch to a branch named **[branch]**

git clone [url]

Download a git repository from **[url]**

branch **[b]**

git remote add [name] [url]

Add a remote repository with **[url]** and an alias of **[name]**

git merge [branch]

Merge branch named **[branch]** with the current one

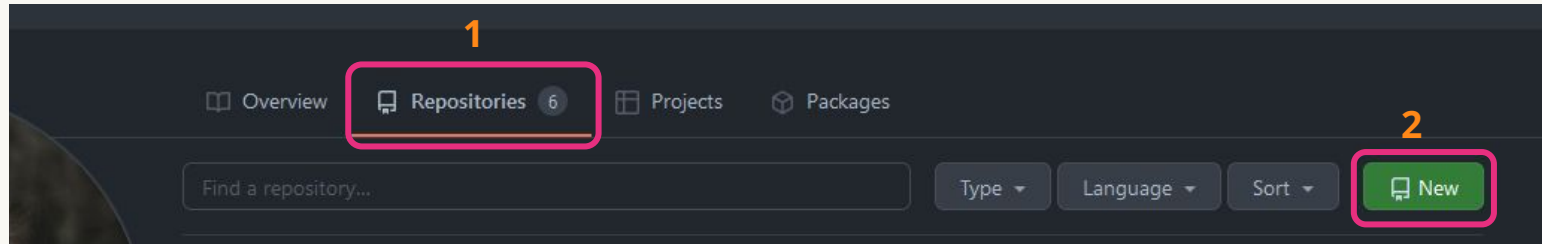
git log

List all the commits in the current branch's history

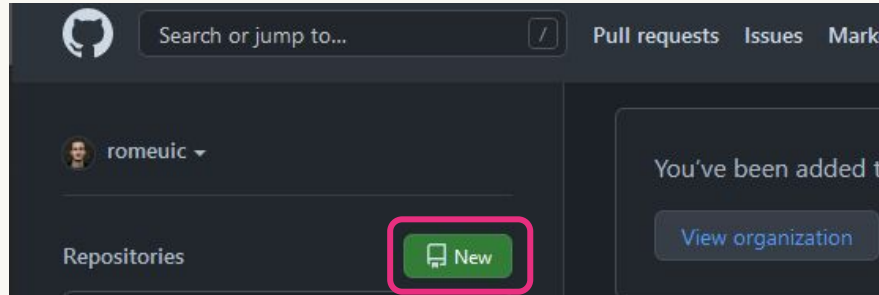
- Vamos criar um repositório do **GitHub**.
- Existem várias formas de criar um repositório novo, mas vamos pelo mais fácil: pelo site do GitHub.
 - **Lembre** de marcar para criar junto do repo um arquivo **README.md**

- <https://github.com/new>

ou





ou



Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 romeuic / exemplo-aula 

Great repository names are short and memorable.

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.


Initialize this repository with:


Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

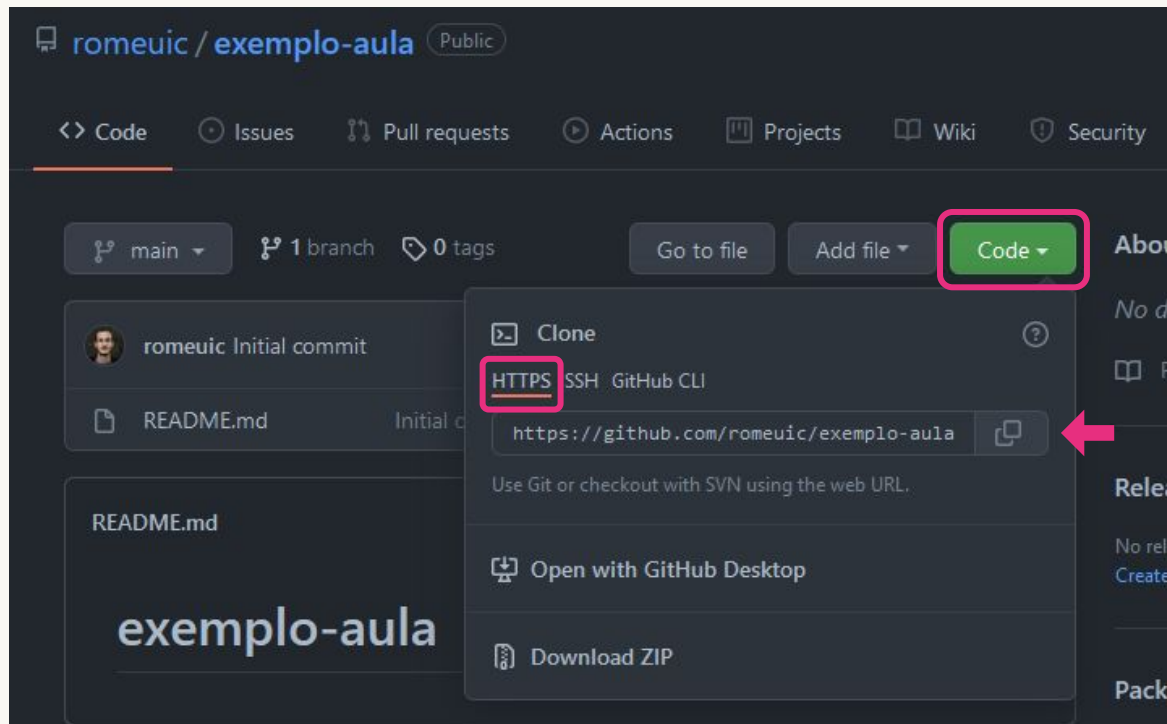
☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

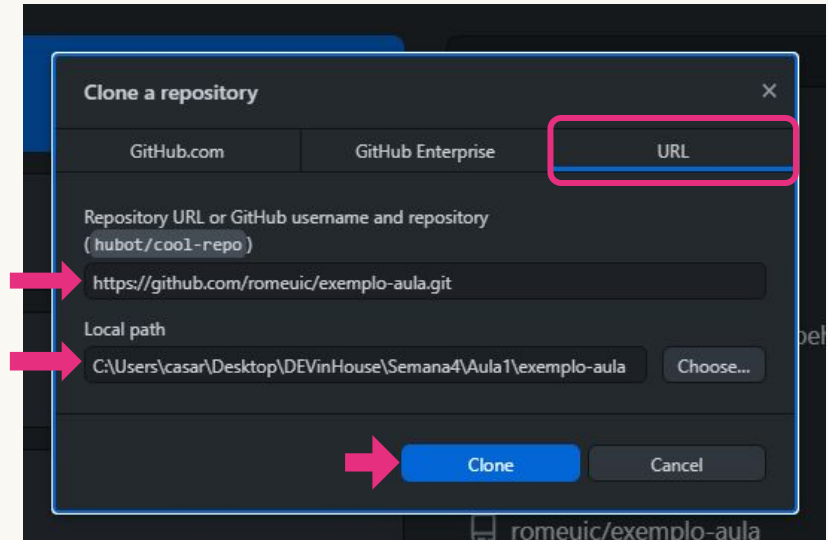
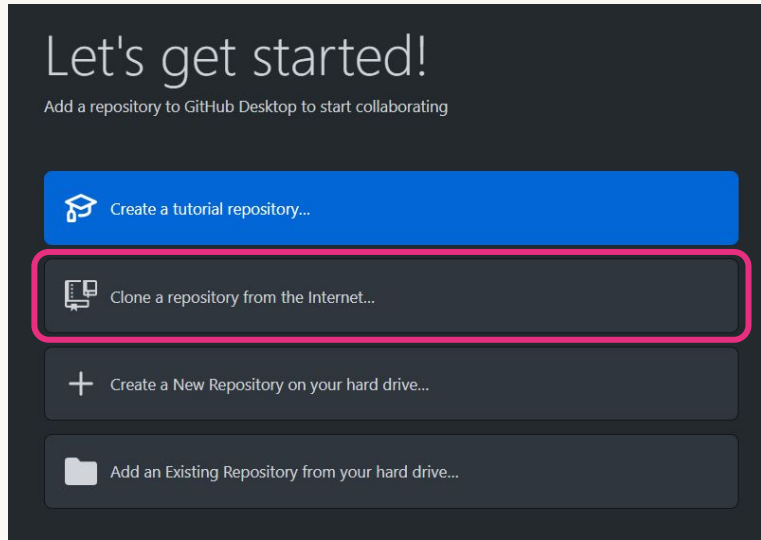
[Create repository](#) 

Escolha se deseja deixar seu repositório visível para qualquer um (**public**), ou apenas para quem você convidar (**private**)

GITHUB | Clone

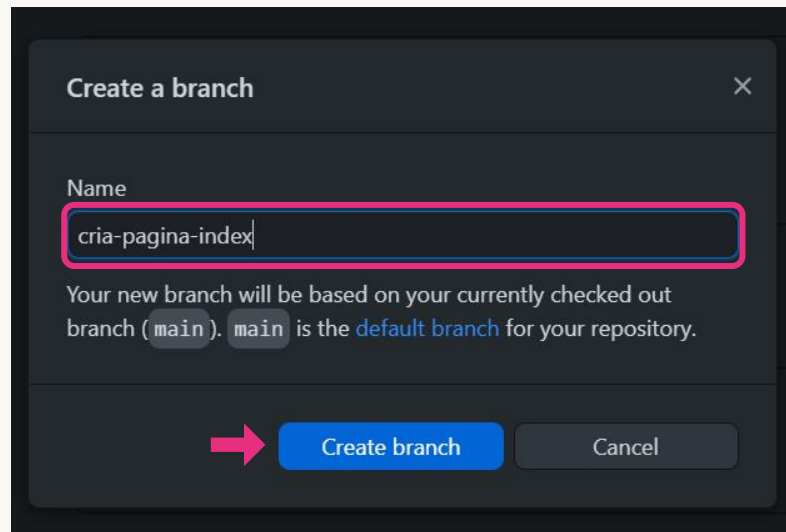
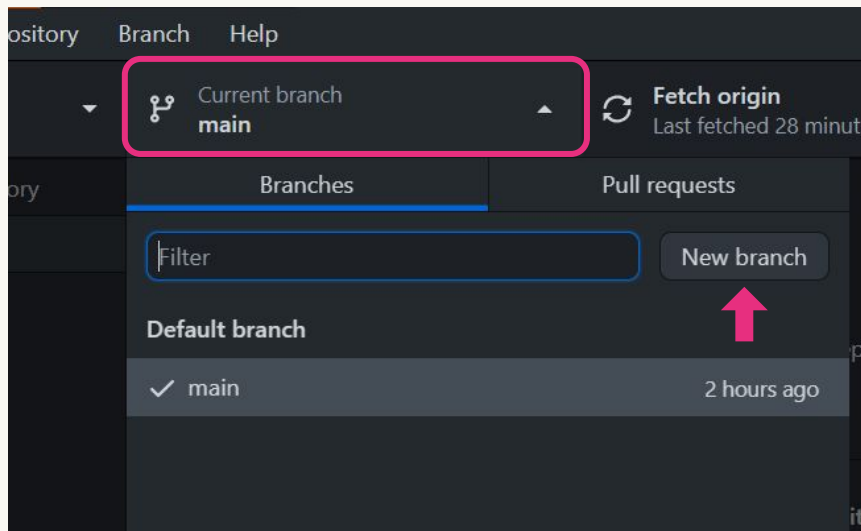


GITHUB DESKTOP | Clone

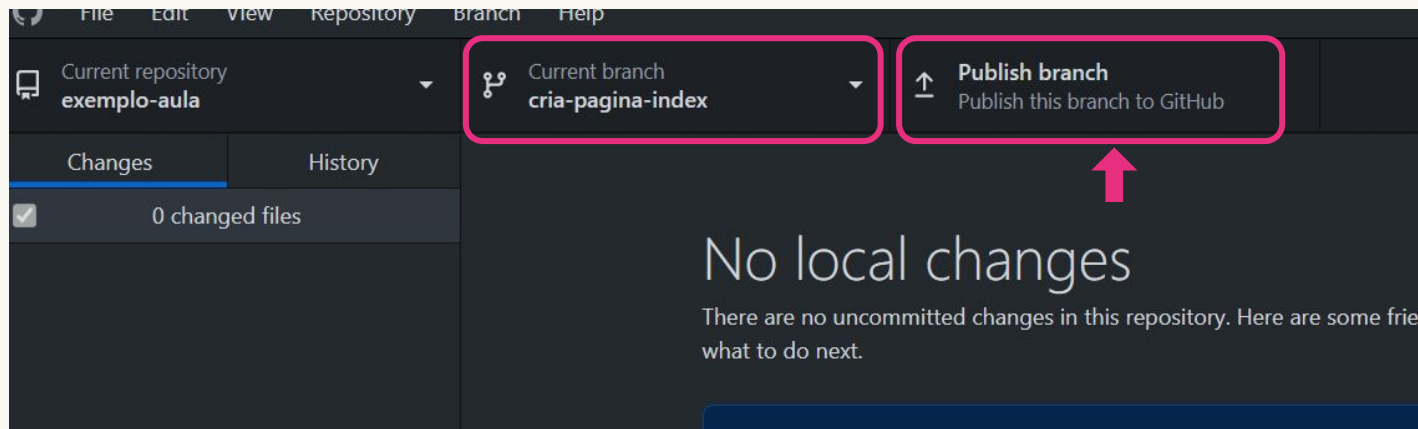


GITHUB DESKTOP | New Branch

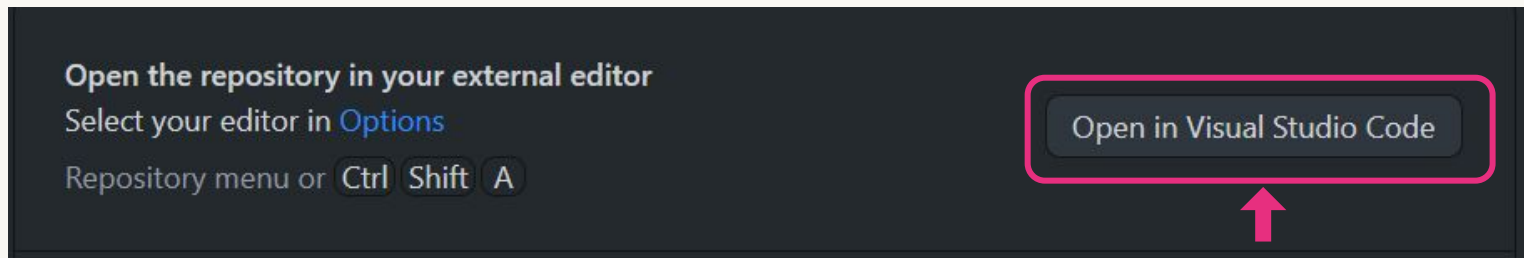
- Vamos criar uma nova **branch**, para inserir um arquivo novo no projeto



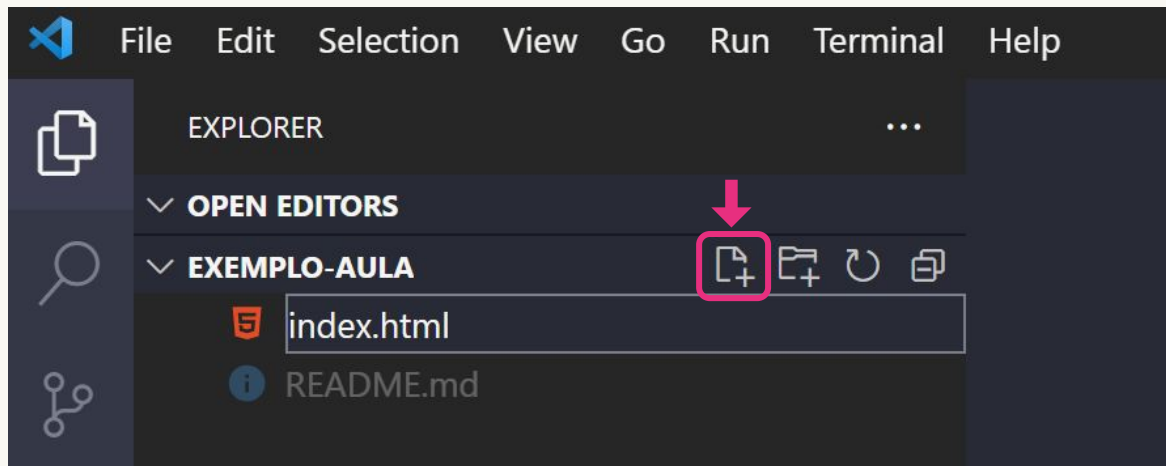
- Nossa **branch** atual está apenas no nosso computador.
- Precisamos informar ao repo remoto (lá no servidor do GitHub) a criação dessa branch. Ou seja, vamos publicá-la.



- Agora vamos fazer nosso trabalho na **branch** atual.
- Vamos abrir o nosso editor de código.
- O próprio GitHub Desktop já nos oferece essa opção.
- Mas sempre podemos abrir independentemente.

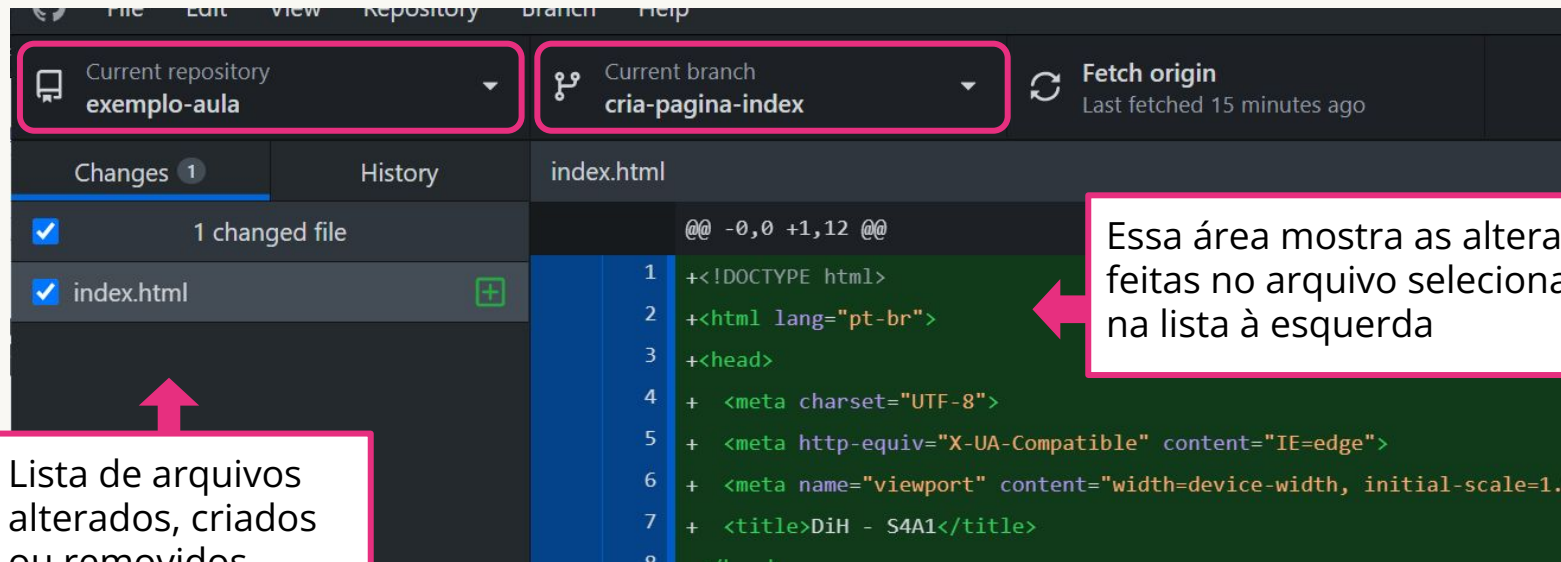


- No nosso editor favorito, podemos criar o arquivo **index.html** e escrever o que precisamos para completar nossa tarefa



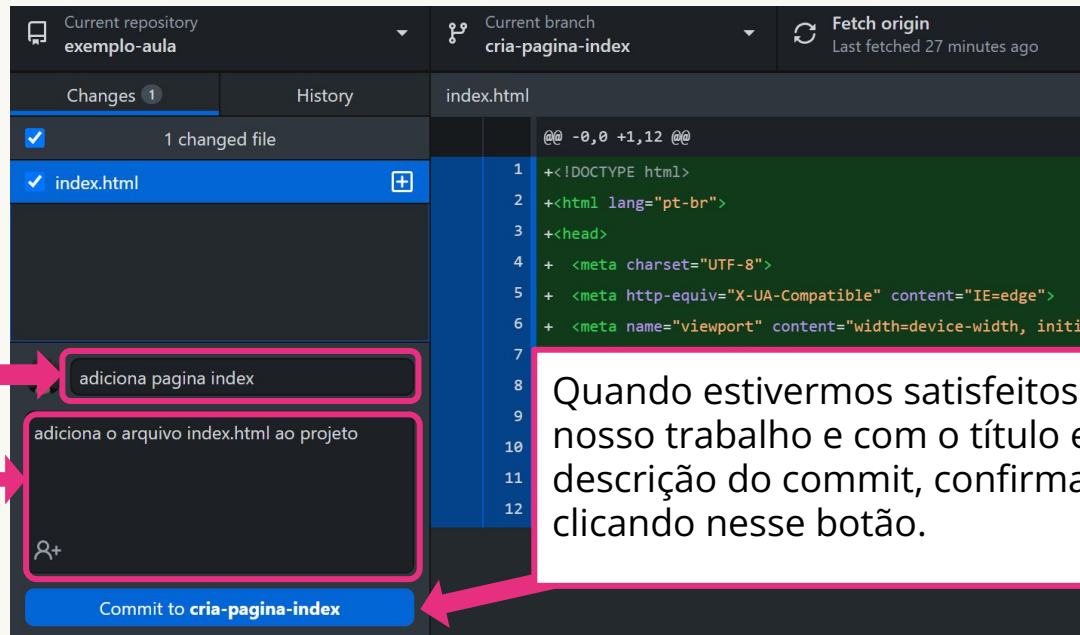
GITHUB DESKTOP | Alterações

- A medida que vamos adicionando arquivos e escrevendo nesses arquivos, essas alterações vão aparecendo no GitHub Desktop.



GITHUB DESKTOP | Commit

- Depois de escrevermos nosso código e finalizarmos a tarefa, salvamos nosso arquivo e voltamos no GitHub Desktop para criar o nosso **commit**.



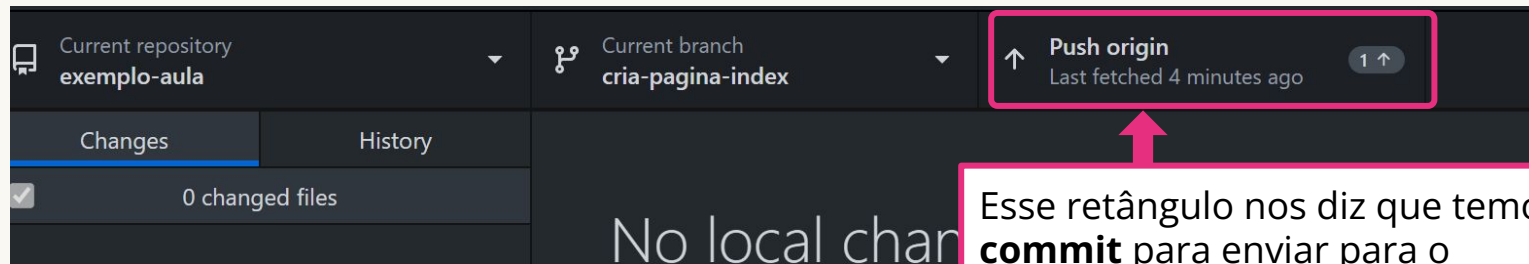
Escrevemos um título para o nosso commit (**obrigatório**)

Podemos escrever uma descrição mais detalhada sobre o que foi feito (**opcional**)

Quando estivermos satisfeitos com nosso trabalho e com o título e descrição do commit, confirmamos clicando nesse botão.

GITHUB DESKTOP | Push

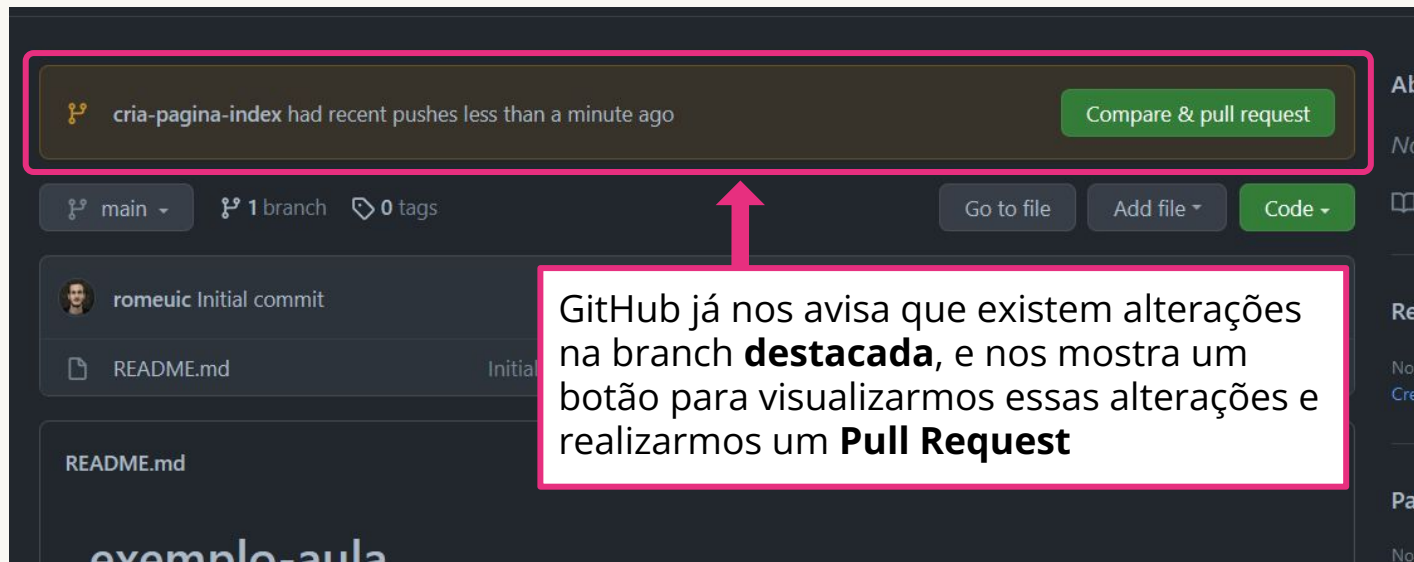
- Fizemos nosso commit, ou seja, salvamos nosso trabalho em um "pacote de alterações".
- Mas ele está salvo apenas no nosso computador.
- Precisamos enviar esse commit, ou esse "pacote de alterações", para o servidor remoto (nesse caso, GitHub).



Esse retângulo nos diz que temos 1 **commit** para enviar para o repositório "**origin**", ou seja, o repositório no servidor do GitHub.

GITHUB | Pull Request

- Depois de enviar nossas alterações para o repositório remoto, vamos ver se está tudo lá no GitHub.



GITHUB | Pull Request

- Ao clicar naquele botão, podemos visualizar as alterações realizadas e escolhermos abrir um PR ou não.

The screenshot shows the GitHub 'Open a pull request' page. At the top, it says 'Open a pull request' and 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).' Below this, there are two dropdown menus: 'base: main' and 'compare: cria-pagina-index'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' The main content area shows a commit titled 'adiciona pagina index' with a description 'adiciona o arquivo index.html ao projeto'. On the right, there are sections for 'Reviewers', 'Assignees', 'Labels', and 'Projects'. A green button at the bottom right says 'Create pull request'.

branch destino (base)

branch origem do Pull Request

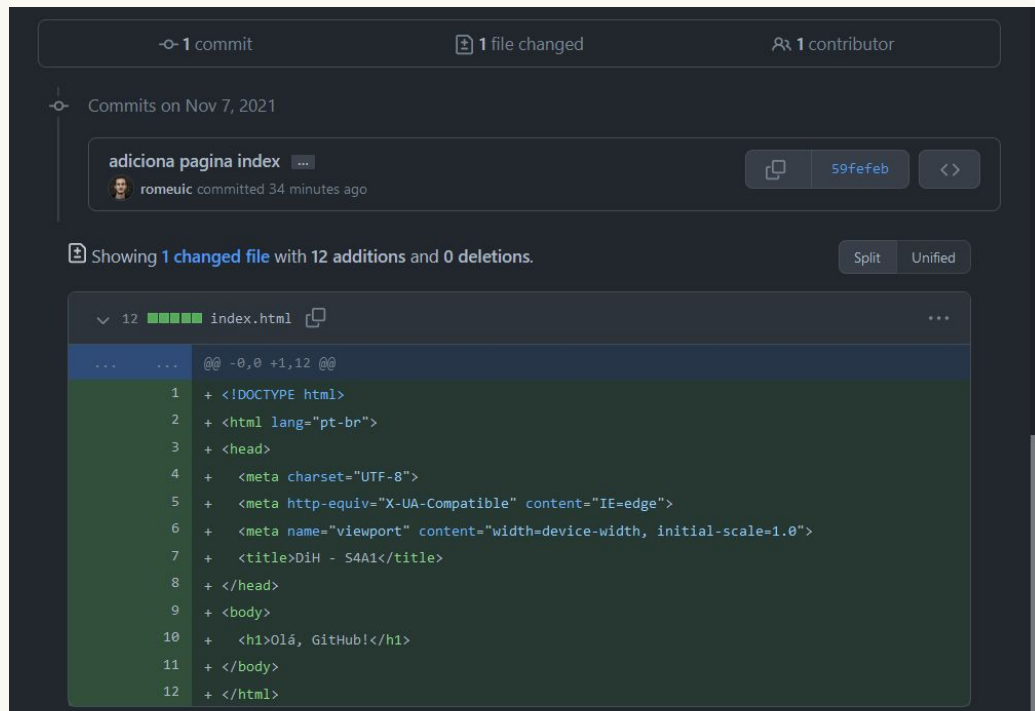
informações relevantes de responsáveis e rótulos do **Pull Request**

role para baixo para ver as alterações

Create pull request

GITHUB | Pull Request

- Podemos ver o que foi alterado nos **commits** dessa **branch**.



1 commit 1 file changed 1 contributor

Commits on Nov 7, 2021

adiciona pagina index ...
romeuic committed 34 minutes ago

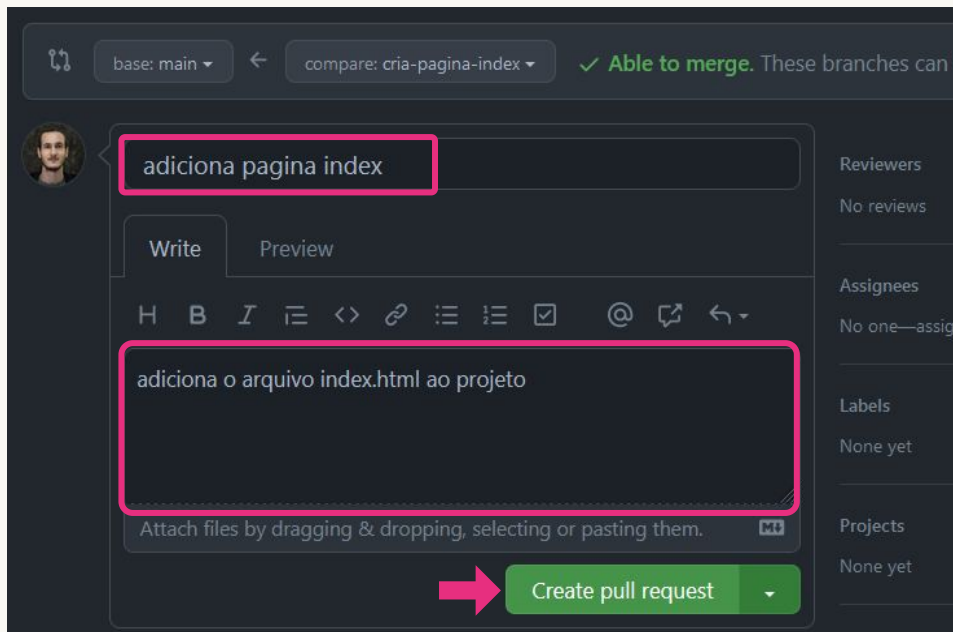
Showing 1 changed file with 12 additions and 0 deletions. Split Unified

12 index.html

```
@@ -0,0 +1,12 @@
1 + <!DOCTYPE html>
2 + <html lang="pt-br">
3 + <head>
4 +   <meta charset="UTF-8">
5 +   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 +   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 +   <title>DiH - S4A1</title>
8 + </head>
9 + <body>
10 +   <h1>Olá, GitHub!</h1>
11 + </body>
12 + </html>
```

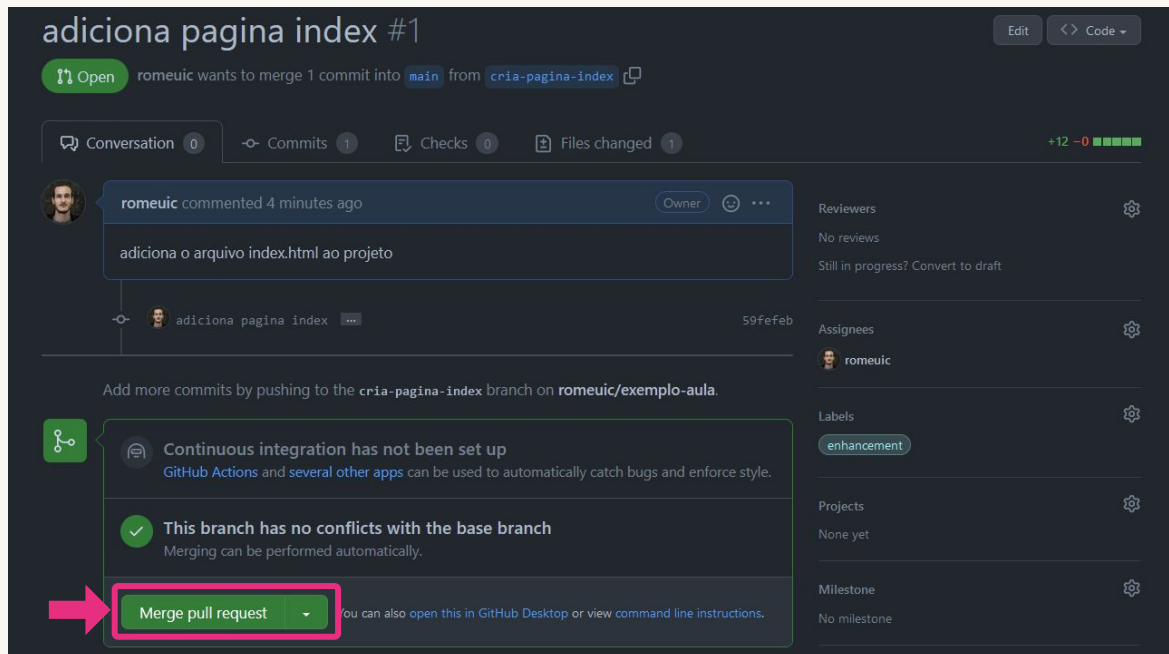
GITHUB | Pull Request

- Após ver as alterações, se estiver tudo OK, podemos alterar o **título** e a **descrição** do nosso **Pull Request**, e então criá-lo.



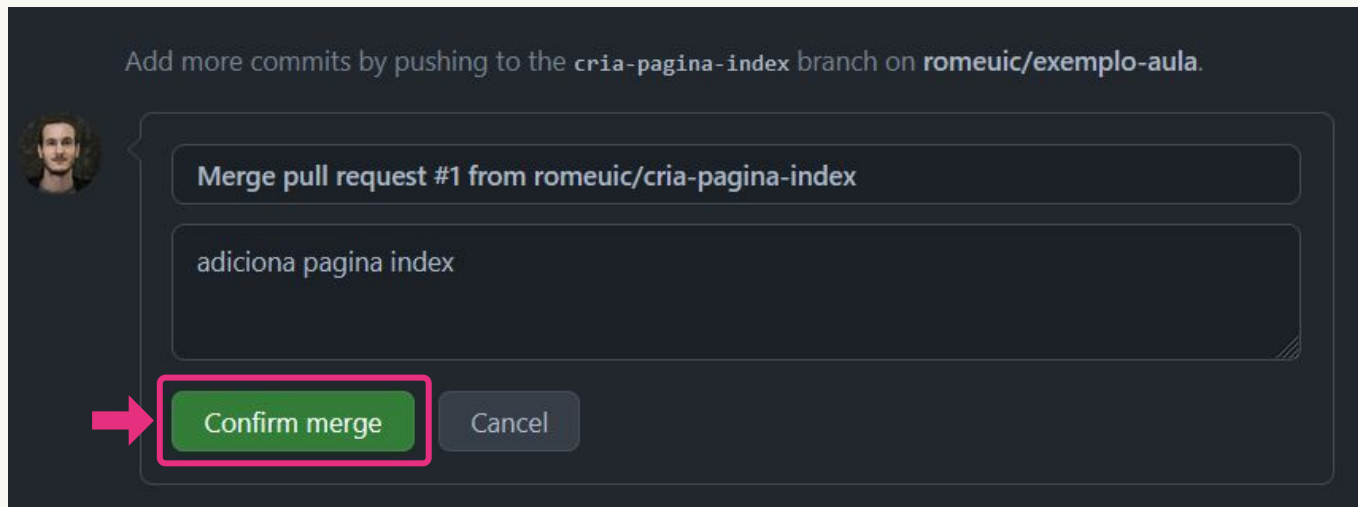
GITHUB | Merge Pull Request

- Com o PR criado, outros(as) colaboradores(as) do projeto podem revisá-lo e fazer comentários, aceitar ou recusar o **merge**.

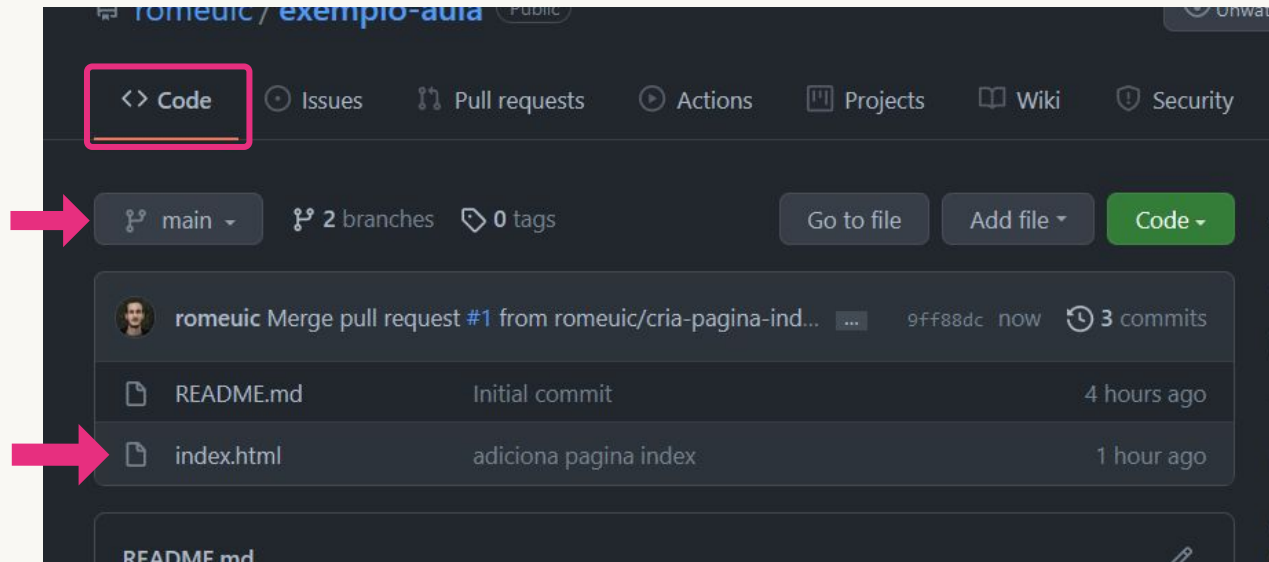


GITHUB | Merge Pull Request

- Após clicar para fazer o **merge**, o GitHub nos pede uma confirmação e podemos alterar a mensagem do **commit** de **merge**.

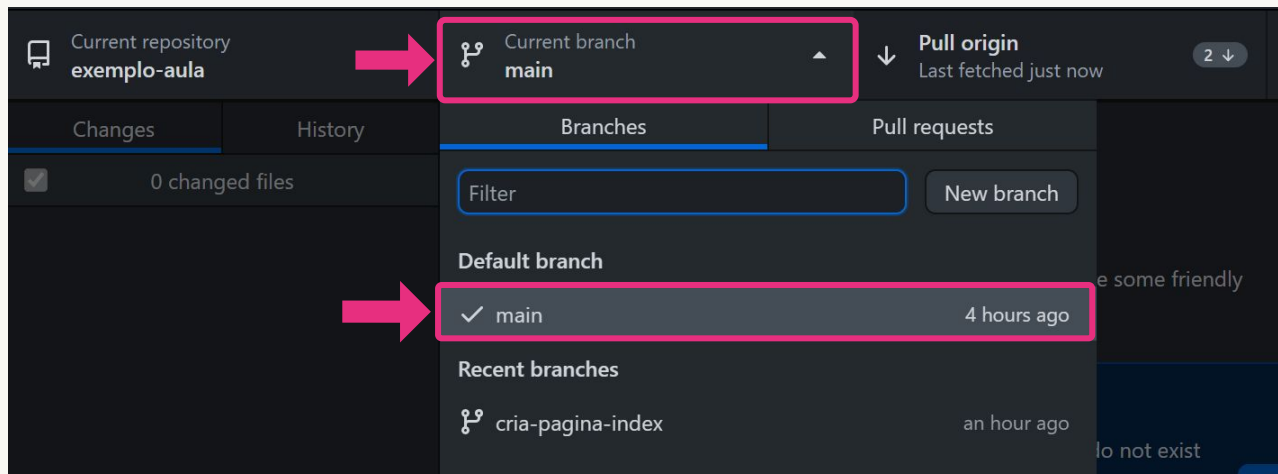


- Retornamos à aba do conteúdo do nosso repositório e vemos que os arquivos que criamos na outra **branch**, agora estão presentes na **branch** principal.

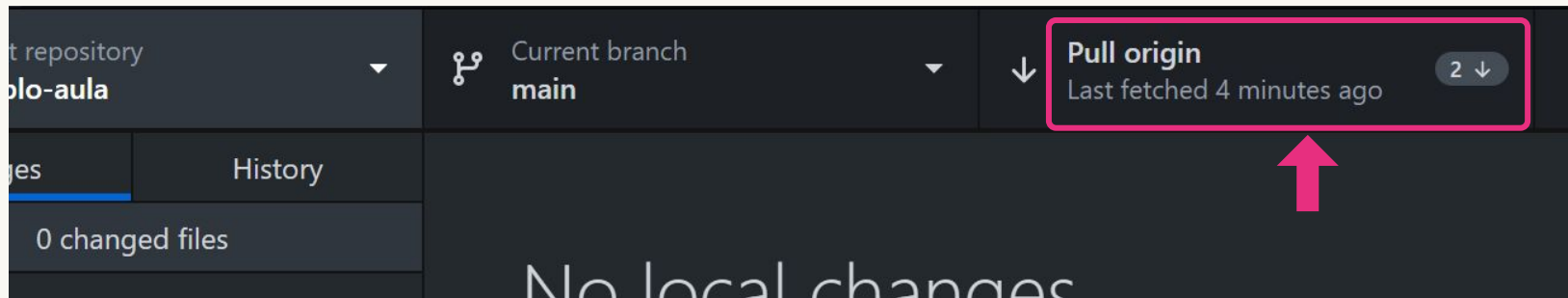


GITHUB DESKTOP | Pull

- Voltando ao GitHub Desktop, vamos atualizar nosso repositório **local**, o que está no nosso computador.
- Primeiro mudamos para a **branch "main"**, pois é onde nosso código está agora.

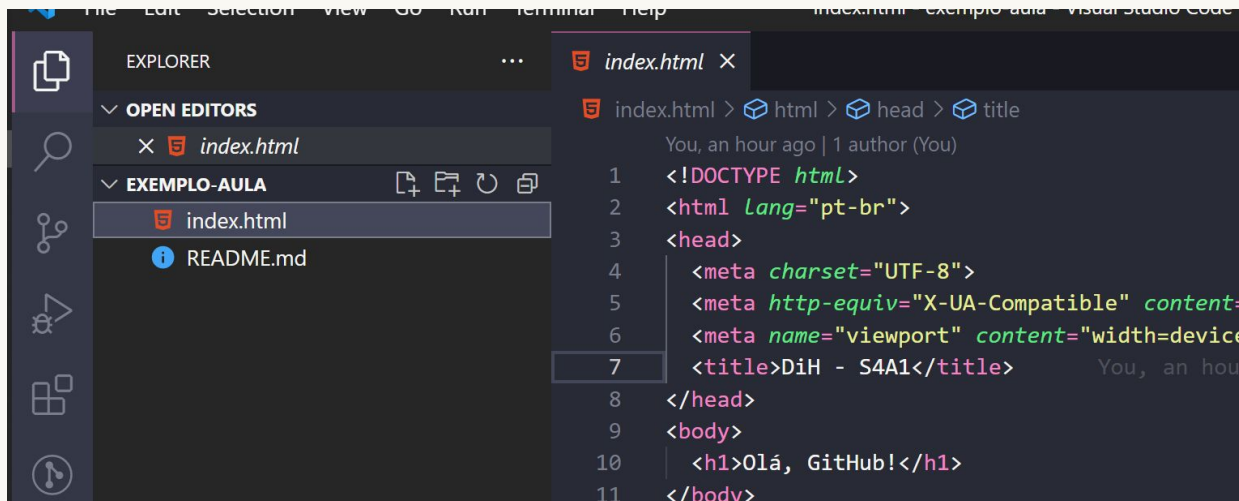


- Então clicamos no botão **"Pull origin"** para baixar a versão mais atualizada do que está no repositório **remoto** (github.com) para o nosso computador.



VS CODE

- Estamos olhando para a **branch "main"**, e o nosso computador está atualizado com a versão mais recente do repositório.
- Quando conferimos os arquivos do nosso projeto, vemos que está tudo atualizado na **branch "main"**.

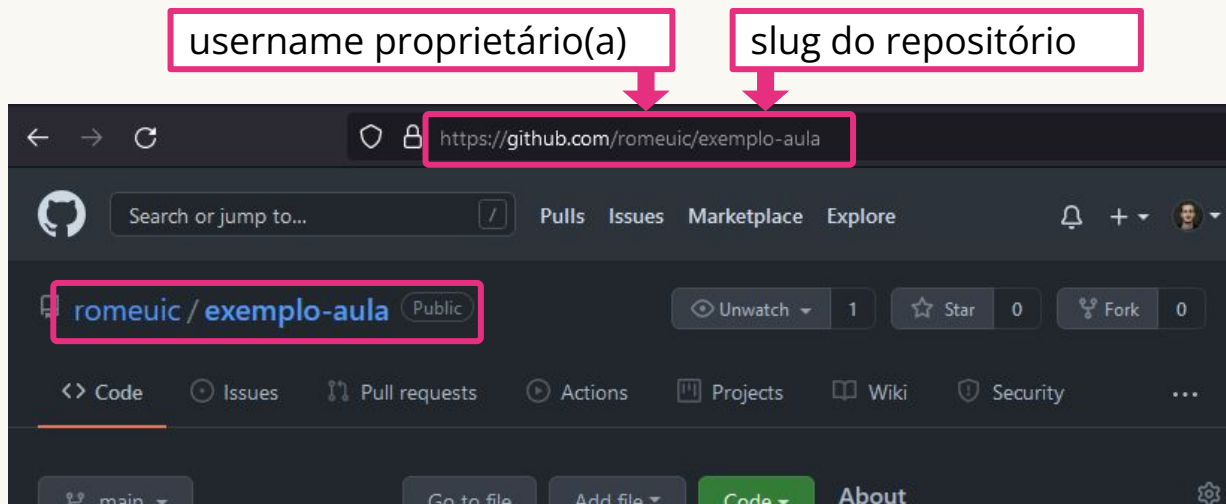


The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar is open, showing the 'EXEMPLO-AULA' folder with files 'index.html' and 'README.md'. The 'index.html' file is selected. The main editor area displays the content of 'index.html', which is an HTML document. The breadcrumb navigation at the top of the editor shows 'index.html > html > head > title'. The code in the editor is as follows:

```
1 <!DOCTYPE html>
2 <html Lang="pt-br">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content=
6   <meta name="viewport" content="width=device
7   <title>DiH - S4A1</title>
8 </head>
9 <body>
10   <h1>Olá, GitHub!</h1>
11 </body>
```

GITHUB | Repositório

- E se você quiser pegar o link desse repositório e enviar no AVA para avaliação?
- Vamos supor que esse repositório é onde está o código do Projeto 1 do curso.



ATRASSO & INTERVALO



- **Timeout** (atraso):

Serve para definir um tempo de atraso na execução de uma função

```
setTimeout(funcao, 1000); //tempo em milissegundos (ms)
```

- **Interval** (intervalo):

Definir um tempo de intervalo entre execuções recorrentes da função

```
setInterval(funcao, 2000); //a cada 2 segundos (2000ms)
```

- **setTimeout**(function, number):

Executa uma função, uma vez, depois de um determinado tempo.

Pode executar uma função que se encontra no próprio arquivo Javascript ou uma referência de uma função definida em outro lugar.

O número representa o intervalo de tempo em milissegundos (1000 milissegundos equivalem a 1 segundo), para esperar antes de executar o código.

Pode ser cancelado utilizando a função **clearTimeout**.

- Exemplo de setTimeout:

```
function olaAtrasado() {  
  console.log('Olá atrasado!');  
}  
  
setTimeout(olaAtrasado, 2000);  
// atraso de 2 segundos (2000ms)
```

- Exemplo de clearTimeout:

```
// guardando a referência do timeout em uma variável  
var t01a = setTimeout(olaAtrasado, 2000);  
  
// cancelando o "agendamento" da execução  
clearTimeout(t01a);
```

- **setInterval(function, number):**

Executa chamadas de funções ou trechos de código específicos repetidamente, de forma recorrente, com um intervalo fixo entre cada chamada (definido em milissegundos).

Para cancelar a execução desta função, basta utilizar a chamada **clearInterval**.

- Exemplo de setInterval:

```
function olaOutraVez() {  
  console.log('Olá outra vez!');  
}  
  
setInterval(olaOutraVez, 2000);  
// intervalo de 2 segundos (2000ms)
```

- Exemplo de clearInterval:

```
// guardando a referência do interval em uma variável  
var i01a = setInterval(olaOutraVez, 2000);  
  
// cancelando o "agendamento" dos intervalos  
clearInterval(i01a);
```



- **Objetos:** são parecidos com vetores, porém, ao invés de estarmos limitados aos índices numéricos de 0 a N (0, 1, 2 ... N), podemos dar nomes aos índices. Inicializamos um objeto com "{}" ao invés de "[]".
- **Chaves** (Keys): Chamamos os índices de um objeto de chaves. Quando inserimos um item em um objeto criamos um par chave-valor (key-value).
- **O acesso** se dá através de um "." (ponto) após o nome do vetor, seguido do nome da chave.
Ex.1: `pessoa.nome;`
Ex.2: `pessoa['nome'];`

```
// inicia um vetor com 3 itens
var vetor = [26, 33, 42];
// acessa o segundo item do vetor (33)
vetor[1];    // 33

// inicia um objeto com 3 chaves
var objeto = { a: 26, b: 33, c: 42 };

// acessa a segunda chave do objeto
objeto.b;    // 33
objeto['b'];  // 33
```

Exemplo de vetor e objeto

OBJETOS | Exemplos

```
// inicia um vetor vazio
var vetor = [];

// insere item no vetor
vetor.push(42);

// altera 1º item no vetor
vetor[0] = 33;

// acessa o 1º item do vetor
vetor[0]; // 33
```

Exemplo de uso de vetor (array)

```
// inicia um objeto vazio
var objeto = {};

// insere/altera uma chave no objeto
objeto.num = 33;
objeto['num'] = 42;

// acessa a chave "num" do objeto
objeto.num; // 42
objeto['num']; // 42
```

Exemplo de uso de objeto (object)

```
// inicia um vetor de objetos
var vetor = [
  { a: 26, b: 32, c: 42 },
  { a: 55, b: 99, c: 11 }
]

// acessa índice e chave
vetor[1].c      // 11
vetor[0]['a']   // 26
```

Exemplo de uso de vetor de objetos

```
// inicia um objeto com vetores
var objeto = {
  a: [26, 32, 42],
  b: [55, 99, 11]
}

// acessa chave e índice
objeto.a[2]      // 42
objeto['b'][0]    // 55
```

Exemplo de uso de objeto com vetores em suas chaves



- **JSON** (**J**ava**S**cript **O**bject **N**otation) é a representação da estrutura de objetos JavaScript em forma de texto (string), baseada em pares de chave-valor e listas.
- Muito utilizado para enviar dados do back-end para o front-end e vice-versa, manipular estruturas de dados com texto.
- Fácil de ler e entender os dados contidos em um texto **JSON**.
- Processamento leve e de fácil interpretação.

- Embora o **JSON** seja derivado do JavaScript, ele é suportado de forma nativa ou através de bibliotecas na maior parte das principais linguagens de programação.
- O que torna esse formato ideal para troca de dados entre aplicações, mesmo que sejam escritas em diferentes linguagens.
- É o formato favorito para quase todos os serviços web disponíveis publicamente, e também é usado com frequência para serviços web privados.

```
// um objeto javascript
var pessoa = {
  id: 19061209,
  nome: 'Grace Hopper',
  nascimento: '1906-12-09',
  condecoracoes: [
    'Legião do Mérito',
    'M. Vitória da 2ª G.Mundial',
    'M. Presid. da Liberdade'
  ]
};
```

Exemplo de um objeto JavaScript com array dentro

```
// uma string JSOM
var pessoaJSON = `{
  "id": 19061209,
  "nome": "Grace Hopper",
  "nascimento": "1906-12-09",
  "condecoracoes": [
    "Legião do Mérito",
    "M. Vitória da 2ª G.Mundial",
    "M. Presid. da Liberdade"
  ]
}`;
```

Representação da estrutura à esquerda em JSON


```
// um array de objetos javascript
var carros = [
  { ano: 1995, modelo: 'Escort' },
  { ano: 1999, modelo: 'Gol' },
  { ano: 1997, modelo: 'Uno' },
  { ano: 2005, modelo: 'Corsa' },
];
```

Exemplo de um array JavaScript com objetos dentro

```
// uma string JSON
var carrosJSON = `[
  { "ano": 1995, "modelo": "Escort" },
  { "ano": 1999, "modelo": "Gol" },
  { "ano": 1997, "modelo": "Uno" },
  { "ano": 2005, "modelo": "Corsa" },
]`;
```

Representação da estrutura à esquerda em JSON

- Para transformarmos objetos JavaScript em texto JSON e texto JSON em objetos JavaScript usaremos as funcionalidades encontrada na entidade **JSON** disponível no ambiente de execução.
- **JSON.parse(texto)** é utilizado para converter um texto JSON (string) em objeto. A chamada deste método nos retornará um objeto.
- **JSON.stringify(objeto)** é utilizado para converter objetos em texto JSON. A chamada deste método nos retornará um texto JSON.

JSON | Stringify & Parse

```
// transformando um objeto em JSON
var carrosJSON = JSON.stringify(carros);

// carrosJSON terá o seguinte valor
`[
  { "ano": 1995, "modelo": "Escort" },
  { "ano": 1999, "modelo": "Gol" },
  { "ano": 1997, "modelo": "Uno" },
  { "ano": 2005, "modelo": "Corsa" },
]`
```

Convertendo um objeto em JSON string

```
// transformando um JSON em objeto
var carros = JSON.parse(carrosJSON);

// carros terá o seguinte valor
[
  { ano: 1995, modelo: 'Escort' },
  { ano: 1999, modelo: 'Gol' },
  { ano: 1997, modelo: 'Uno' },
  { ano: 2005, modelo: 'Corsa' },
]
```

Convertendo uma JSON string em objeto

MATERIAL COMPLEMENTAR



What is version control? | <https://git-scm.com/video/what-is-version-control>

Vídeo: "O que são Git e GitHub?" (7min) | https://youtu.be/P4BNi_yPehc

Entendendo GIT | <https://youtu.be/6Czd1Yetaac>

Curso JavaScript #46 - setTimeout e setInterval | <https://youtu.be/tXnY9-gVN1E>

Curso JavaScript #47 - clearTimeout e clearInterval | <https://youtu.be/KV1ph8CYWi4>

JSON em Javascript | <https://youtu.be/ziOjvR56qOw>

O que é JSON | <https://youtu.be/K1f7G0JMkLU>

MATERIAL COMPLEMENTAR



Tutorial de Git pelo GitHub (inglês) | <https://guides.github.com/activities/hello-world>

Tutorial de Git pelo Bitbucket (pt-br) | atlassian.com/br/git/tutorials/learn-git-with-bitbucket-cloud

Git Handbook | <https://guides.github.com/introduction/git-handbook>

Documentação Git | <https://git-scm.com/doc>

Referências Git | <https://git-scm.com/docs>

Hello World | GitHub | <https://guides.github.com/activities/hello-world>

setTimeout() | <https://developer.mozilla.org/en-US/docs/Web/API/setTimeout>

setInterval() | <https://developer.mozilla.org/en-US/docs/Web/API/setInterval>

Objeto - JavaScript | https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Object

JavaScript Objects | <https://developer.mozilla.org/en-US/docs/Web/API/setInterval>

Trabalhando com JSON | <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/JSON>



DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!



<LAB365>