

Introdução a Programação Orientação a Objetos I

Código da turma no classroom: iqmtxhd
Aulas: Segunda - 20:20 e Quinta - 18:30

Para aprofundar!

Escolher 1 artigo relacionado ao tema e realizar leitura para a próxima aula

2018 > Atual

Pesquisa no
Google Scholar

Trazer
informações
relevantes e
impressões

Git - Guia Prático

apenas um guia prático para começar com git. sem complicações ;)

O QUE TEREMOS NA AULA DE HOJE!



ASPECTOS
INICIAIS

TRABALHANDO COM
REPOSITÓRIOS

FLUXO DE
TRABALHO

COMANDOS
BÁSICOS

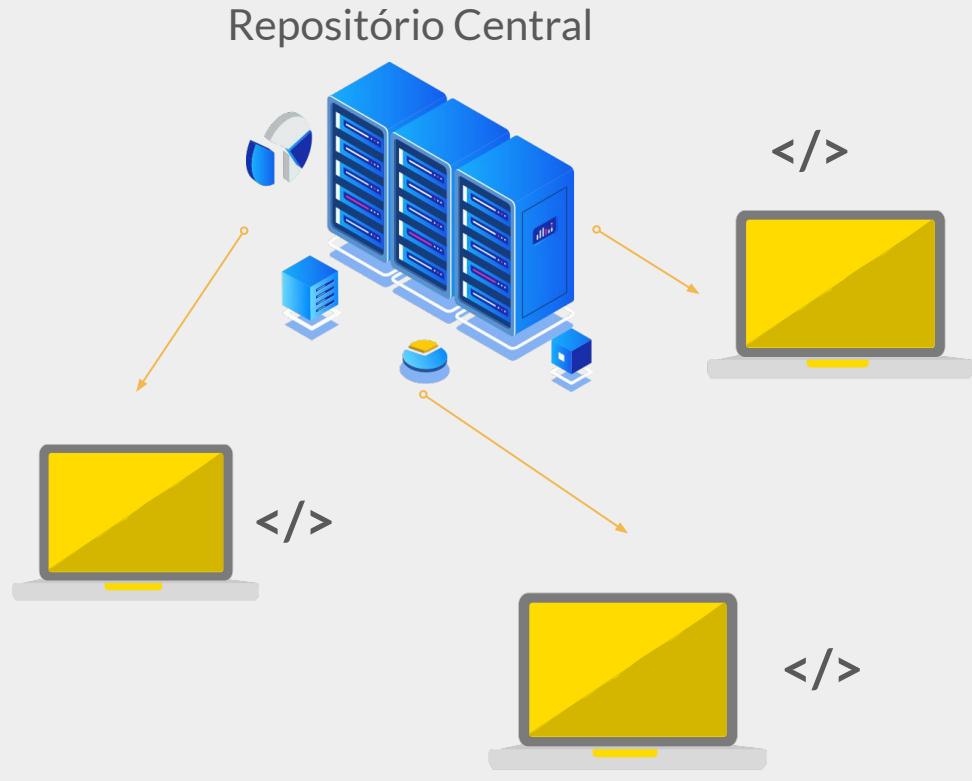
PRÁTICA

Motivação

- Sistemas de Controle de Versão
 - “Controle de versão é um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa lembrar versões específicas.”
- O que pode ser versionado?
 - Praticamente tudo, exemplos: código fonte, documentos, produções de diversos tipos.

Sistemas de Controle de Versão Centralizados

No controle de versão centralizado há um único repositório e várias cópias de trabalho que se comunicam apenas através do repositório central.

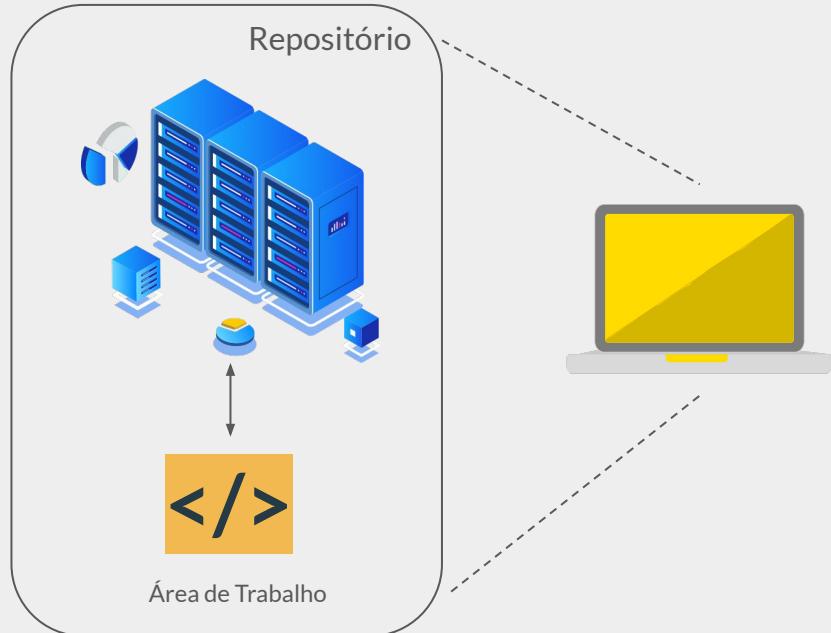


- **Commit:** envia para o servidor
- **Update:** atualiza local

Sistemas de Controle de Versão Distribuídos

No controle de versão distribuído cada desenvolvedor possui um repositório próprio acoplado a uma área de trabalho.

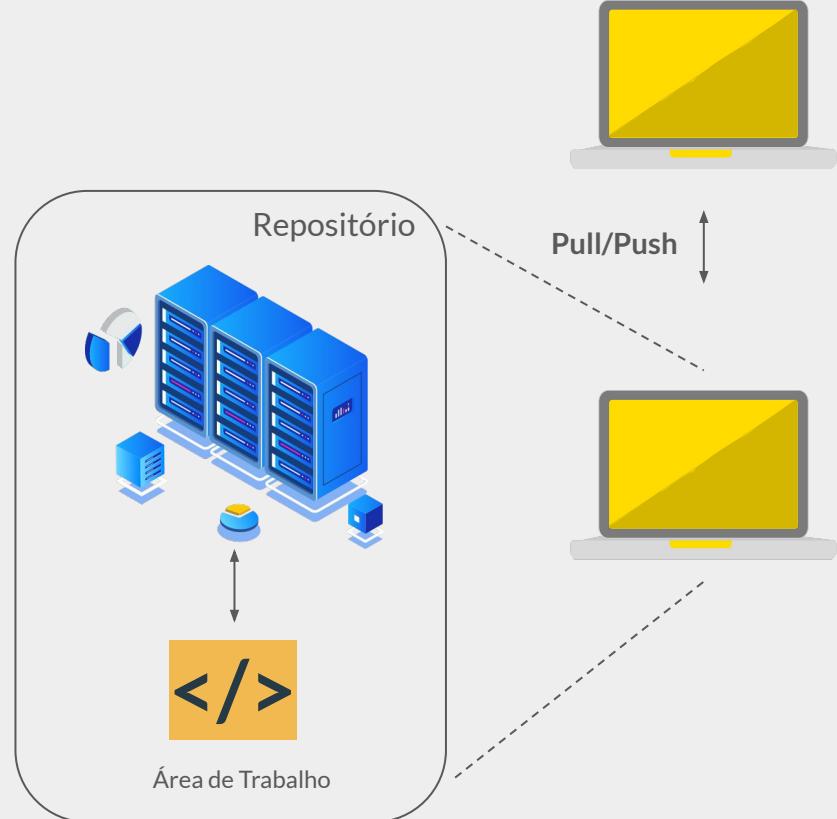
- **Commit:** envia para o servidor
- **Update:** atualiza local



Sistemas de Controle de Versão Distribuídos

Um repositório recebe e envia revisões com qualquer outro através de operações pull e push sem a necessidade de uma topologia pré-definida.

- **Pull:** Atualiza o repositório local (destino) com todas as alterações feitas em outro repositório (origem).
- **Push:** Envia as alterações do repositório local (origem) para um outro repositório (destino).



Introdução ao git/github

- São utilizados no dia a dia das pessoas que criam software por um motivo bem simples: ter uma **forma fácil de gerenciar** o código fonte da aplicação, do sistema, do produto.





O que é Git?

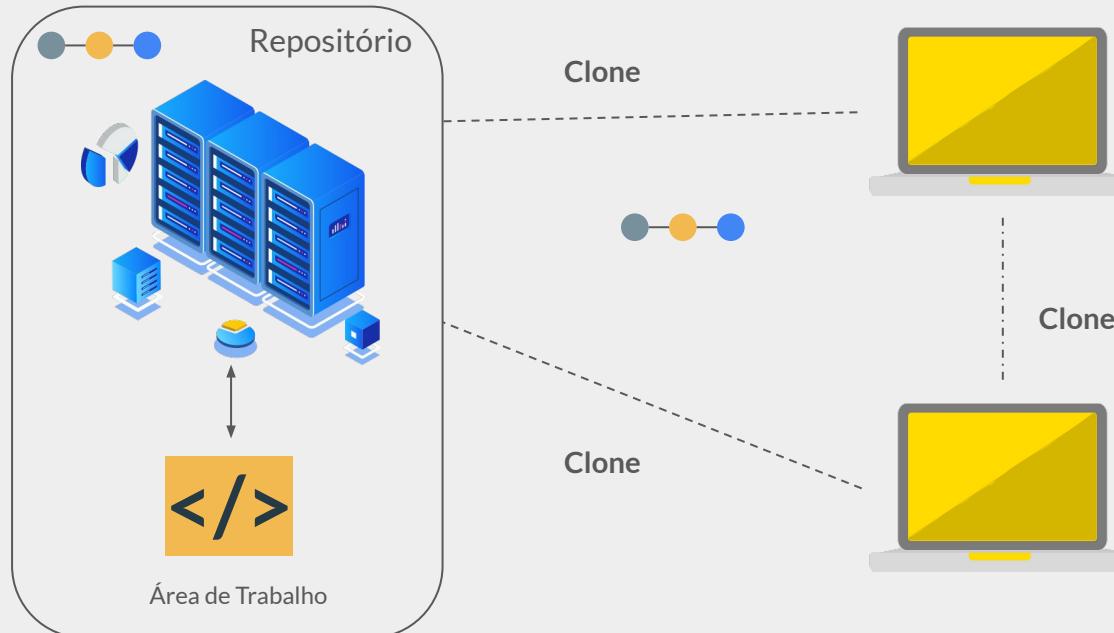
- O Git é um **sistema de controle de versão** distribuído e amplamente adotado.
- O objetivo é acessar o código colaborativo e manter o histórico dos arquivos.
- O Git nasceu e foi tomando espaço dos outros sistemas de controle.



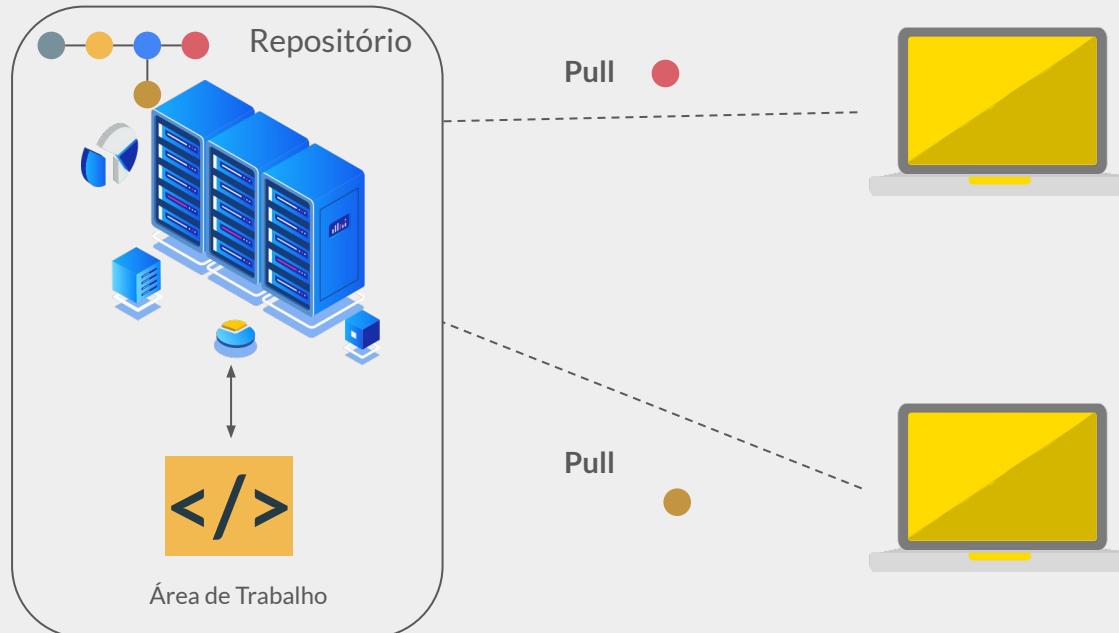
O que é Git?

- GitHub é uma plataforma para **gerenciar seu código e criar um ambiente de colaboração** entre devs, utilizando o **Git** como sistema de controle.
- Objetivo é facilitar uso do Git, escondendo alguns detalhes mais complicados de setup.
- É onde efetivamente você vai ter seu repositório e usar no dia a dia.

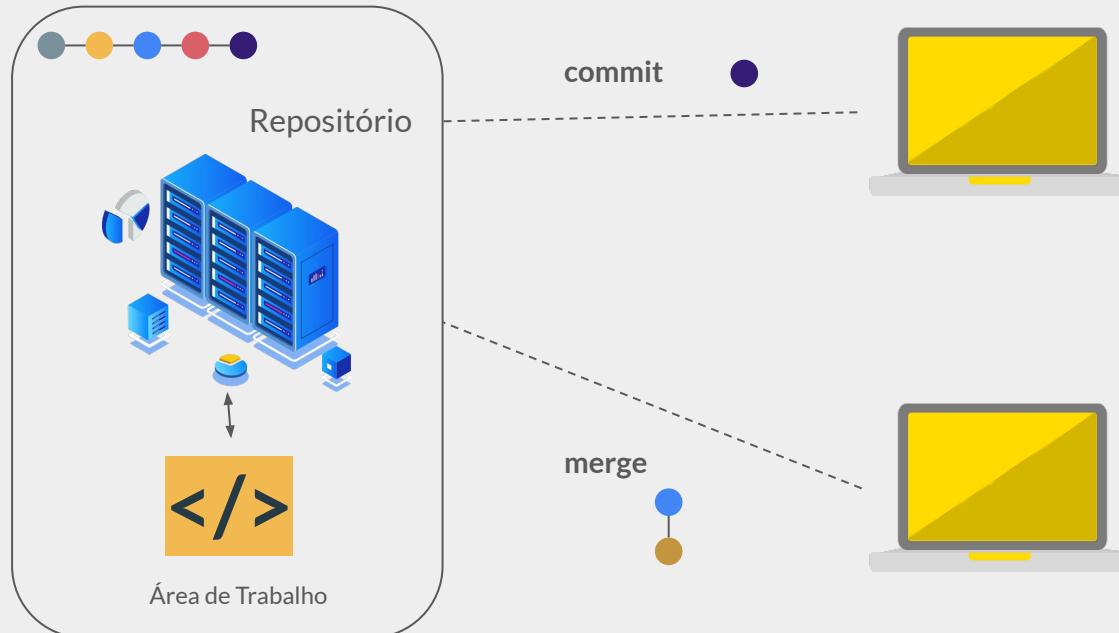
Fluxo: Sistemas de Controle de Versão Distribuídos



Fluxo: Sistemas de Controle de Versão Distribuídos



Fluxo: Sistemas de Controle de Versão Distribuídos





Passos iniciais - Instalando

Git para Windows

Git para macOS

Git para Linux

Comandos Básicos - Git

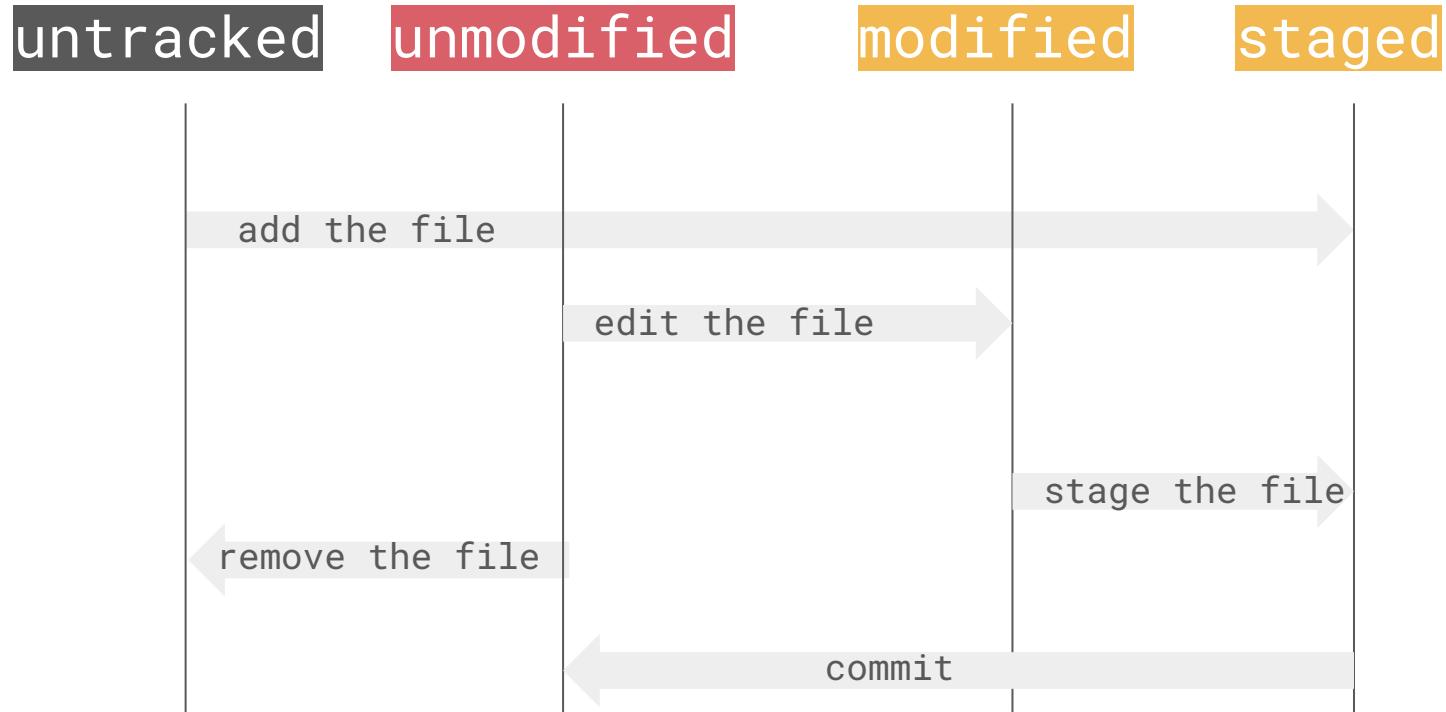
— INTRODUÇÃO A DSW

- Configuração inicial
 - \$ git config --global user.name “**Sarah Soares**”
 - \$ git config --global user.email “**contato@sarahsoares.com.br**”
 - \$ git config --global core.editor nano
- Verificar
 - \$ git config user.name
 - \$ git config --list
- Ajuda
 - \$ git help config

- Inicialização de um repositório
 - `$ mkdir curso-git` //cria um diretório
 - `$ cd curso-git` //ativa um diretório
 - `$ git init` //inicializa o versionamento do diretório
 - `$ ls -la` //lista o conteúdo do diretório
- Informações sobre o repositório (diretório `.git`)
 - `$ cd .git`
 - `$ ls`

**branches config description HEAD hooks info
objects refs**

Ciclo de Vida dos Arquivos - Git



- Criar e alterar um arquivo para ser versionado
 - \$ cd curso-git //diretório controlado pelo git
 - \$ nano Readme.md //pode usar qualquer editor
Ctrl + o //escreve no arquivo
Ctrl + x //encerra o editor
- Ocultar arquivos do versionamento
 - \$ cd .git/info
 - \$ nano exclude //arquivo (file.cpp) ou regra (*.class)

- Verificação do status de um repositório
 - `$ git status`
- Adicionar uma versão de um arquivo para a área transferência
 - `$ git add Readme.md`
- Confirmar uma versão de um arquivo no repositório
 - `$ git commit -m "Add Readme.md"`
 - `$ git status`

- Verificar log do repositório
 - \$ git log
[**--decorate, --graph, --stat, author="Sarah"**]
 - \$ git log --pretty=oneline
 - \$ git shortlog
- Mostrar um commit do repositório
 - \$ git show
0856ee55ba88084595e54dfb5857a55c3407af3a
- Verificar diferenças no repositório
 - \$ git diff

- Desfazer alterações no repositório (modo de edição)
 - \$ git checkout Readme.md
 - \$ git diff
 - \$ git status
- Desfazer alterações (modo de transferência)
 - \$ git reset HEAD Readme.md
- Desfazer alterações (modo de versões)
 - \$ git reset [--soft, --mixed, --hard] hash

Repositórios Remotos – GitHub

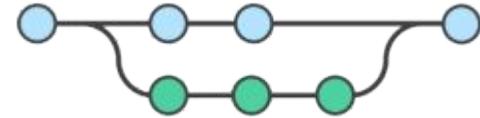
- Criar ou acessar uma conta (github.com)
- Criar um repositório (user/repository)
- Enviar as versões para o repositório remoto
 - \$ git remote add origin
https://github.com:user/repository.git
 - \$ git push -u origin master
 - \$ Username for 'https://github.com': user
 - \$ Password for
'https://user@domain@github.com': password

Repositórios Remotos – GitHub

- Criar ou acessar uma conta (github.com)
- Criar um repositório (user/repository)
- Criar uma chave de acesso
(<https://help.github.com/en/articles/about-ssh>)
 - \$ cd ~/.ssh (no linux)
 - \$ ssh-keygen -t rsa -b 4096 -C "user@domain.com"
- Adicionar chave nas configurações do GitHub
- Enviar as versões para o repositório remoto
 - \$ git remote add origin <git@github.com:user/repository.git>
 - \$ git push -u origin master

Repositórios Remotos – GitHub

- Mostrar repositório remoto
 - \$ remote show origin
- Renomear repositório remoto
 - \$ git remote rename origin origin-new
- Remover repositórios remotos
 - \$ git remote remove origin-new
- Clonar repositórios remotos
 - \$ git clone git@github.com:user/repository repository-clone
 - \$ cd repository-clone \$ cat Readme.md



Branches

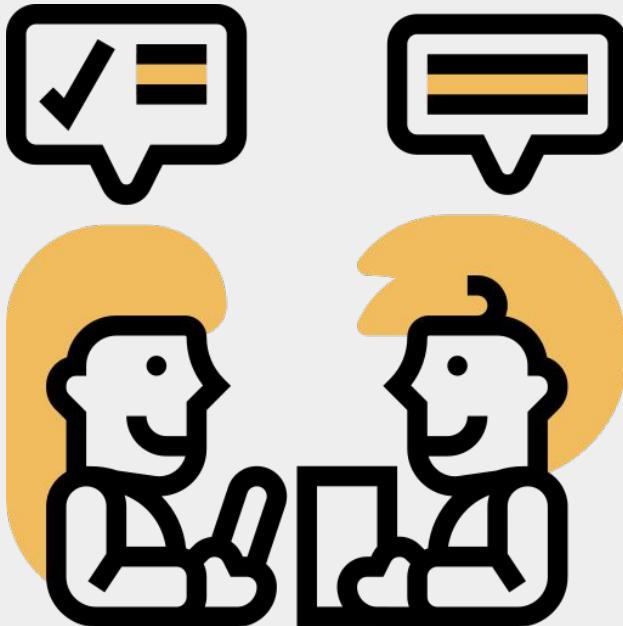
- Por que usar branches no git?
 - É leve e facilmente desligável
 - Modificar os arquivos sem alterar o branch master
 - Permite o trabalho de múltiplas pessoas no projeto
 - Evita alguns conflitos
 - Fornece ferramentas para gerenciar outros conflitos

Formas de usar!

- O Github Desktop é uma **casca para o comando do git**, que esconde alguns termos que podem ser assustadores para quem está começando.
- Eu indico fortemente **seu uso como primeiro passo**, pois o sistema inteiro dá uma cara de "sincronizador de código".
- Facilita a visualizações, o envio e recebimento das modificações e os famosos conflitos de merge, que **você não precisa se preocupar nesse primeiro instante**.

Projeto Prático

- **Primeiro passo:** criar o repositório github do seu projeto e adicionar uma breve descrição da sua ideia.
- **Enviar:** enviem o link via classroom.



Para aprofundar!

Acessem os links na caixinha.

Acessar a documentação do Github: ver aspectos gerais, recursos e instruções iniciais

Solicitar GitHub Student Developer Pack

Série de vídeos sobre git e github

Introdução a Programação Orientação a Objetos I

Código da turma no classroom: iqmtxhd
Aulas: Segunda - 20:20 e Quinta - 18:30

sarahsoares.com.br/