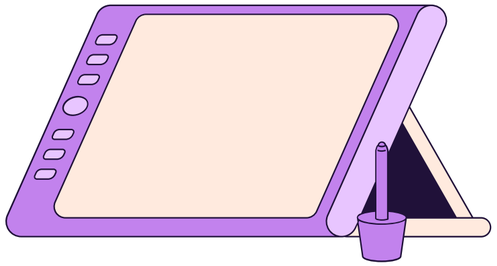
Versionamento de código

Professora Tabita Barbosa

Exercício para semana

- Faça um clone no seu pc -> https://github.com/reprograma/On14-TodasEmTech-s1-VC.git
- Entrar na pasta do clone
- Cria uma branch com seu nome
- Crie um arquivo com seu nome e escreva uma música, um lugar e um livro ou disco favorito seu
- Adicione esse arquivo ao olhar do git
- Faça um commit
- Faça um git push para repositório
- No seu repositório no GitHub faça um Pull Request, ou seja envie a sua branch para a branch do repositório original



Arranjo da aula

- Visão Geral da área de desenvolvimento e conversa
  - **Intervalo 15 minutos**
- O que é versionamento de código?
- Ferramentas
  - **Almoço 1h**
- Comandos básicos
- Configuração do Git
  - **Intervalo 15 minutos**
- Exercícios na aula
- Exercício para entrega

Linha de comando - básicos

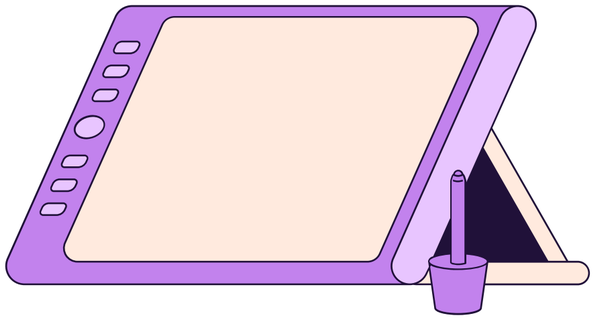
- Interfaces que interpretam linhas de comando, consoles:
  - Command Prompt / cmd -> interpretador padrão do windows, simples e funcional
  - PowerShell -> mais robusto, é uma ferramenta mais recomendada pra usuários com capacidade de programação mais avançada
  - Bash -> criado como software livre, é um Unix shell e linguagem de comando, assim como o anterior
- Comandos básicos que podemos executar através da linha de comando, momento matrix!
  - pwd -> diretório/pasta atual. mostra o caminho todo, ex: tabitaPc/documentos/dev/reprograma
  - ls -> lista arquivos ou pastas presentes na pasta que você esta
  - cd nome-da-pasta -> muda a pasta, entra em outra pasta com o nome dado
  - cd ~ -> volta a pasta raiz
  - cd .. -> volta uma pasta atrás
  - mkdir nome-da-pasta -> cria uma pasta

Linha de comando - básicos

  - rm nome-do-arquivo -> deleta um arquivo
  - rm -f ou rm --recursive nome-da-pasta -> deleta uma pasta
  - whoami -> identifica usuário que esta logado
- Duas formas pra criar e editar um txt pela linha:
  - echo Oi Bonitas > texto.txt -> você adiciona texto pela linha de comando, mas um pouco mais simples
  - cat > meu-texto.txt -> cat abre pra edição do arquivo no próprio console, permite mais linhas e mais complexidade. Para sair do modo edição clique no ctrl + z.

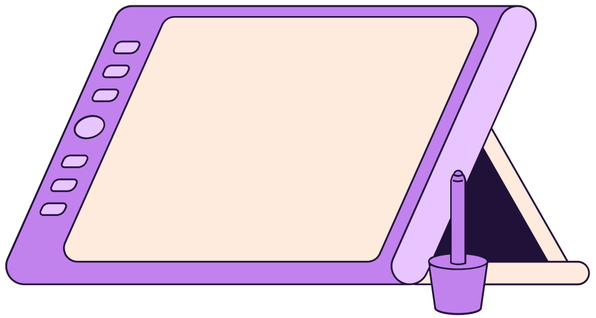
Hora do Exercício

- Abra o Bash
- Identifique o usuário -> whoami
- Confirma a pasta em que esta -> pwd
- Crie uma pasta -> mkdir nome-da-pasta
- Entre na pasta -> cd nome-da-pasta
- Crie um arquivo e insira uma frase inspiradora, ou não rs - pelo terminal!
echo frase > nome.txt
cat > nome.txt + enter e escreve o texto. terminou? ctrl + z
- Tire um print e mostra pra gente!



Hora do Exercício

- Abra o Bash
- Confirma a pasta em que esta -> pwd
- Entre na pasta criada antes -> cd nome-da-pasta
- Apague o arquivo criado -> rm nome-do-arquivo
- Volte uma pasta -> cd ..
- Apague a pasta criada -> rm -f nome-da-pasta



Ferramentas de Versionamento

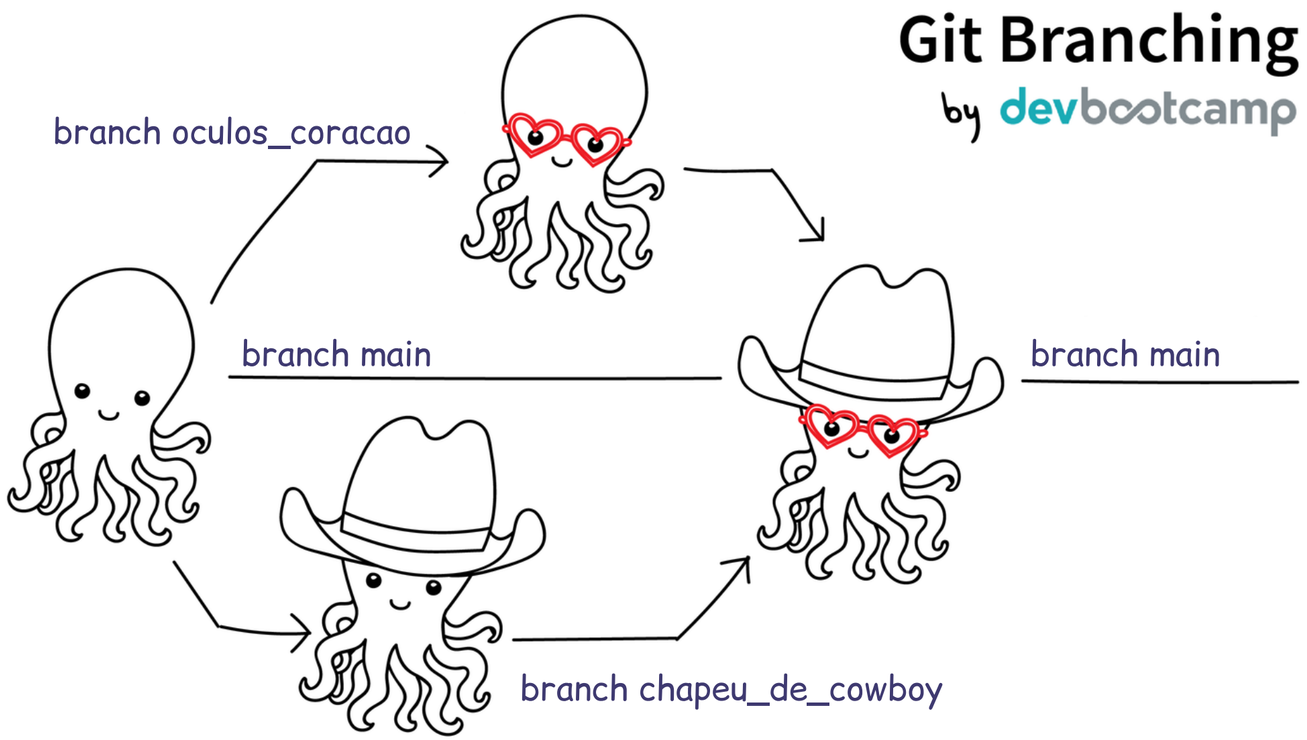
- Existem algumas plataformas onde podemos compartilhar/versionar códigos. Exemplos:

Versionamento de código - Git

- Versionamento é ótimo para trabalho em equipe e solo, através dele podemos registrar as versões de um arquivo, codar em simultanêo a outras pessoas e juntar trabalhos de forma segura e documentada
- Conceitos básicos
  - Repositório -> espaço/pasta onde nosso projeto é salvo
  - Commit (compromentimento) -> cria um "marco" que controla a versão do arquivo, recebe uma mensagem que identifica o que foi alterado pelo desenvolvedor
  - Pull (puxar) -> a partir do repo remoto que estamos ligados puxa as atualizações dele para o nosso repositório local
  - Push (empurrar) -> envia as alterações locais para o repositório remoto
  - Clone -> clona literalmente o repositório que passamos através de url ou ssh
  - Fork: é uma cópia de um projeto para o seu GitHub, é como uma xerox mesmo. É feito pelo github mesmo
  - Pull Request: é um solicitação de merge da sua branch em um projeto de outra pessoa/empresa.
    - Exemplo: Você faz um fork de um projeto, clona a partir da sua url, cria e altera na sua branch local e depois faz um push pro seu repo. Ele dá a opção de fazer um PR, que é como um merge, pode ser feito no seu próprio repo ou entre repositórios. O PR precisa ser aprovado e revisado, e pode receber comentários dos seus pares de equipe. MUITO usado em grupo de trabalho.

Versionamento de código - Git

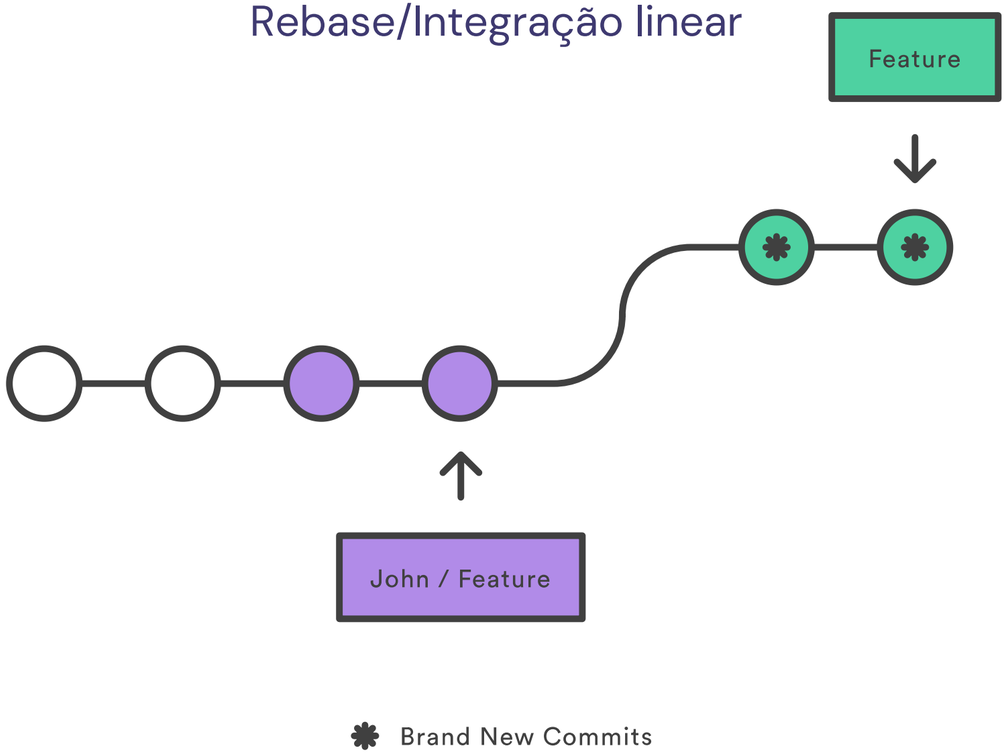
- Branches (galhos) -> a coisa mais legal e útil em um desenvolvimento coletivo. Permite que cada usuário ou funcionalidade sendo desenvolvida tenha seu próprio bracinho dentro do git e possa ser desenvolvida de forma independente
- Merge (fundir) -> quando colocamos diferentes branches em uma mesma. Exemplo:



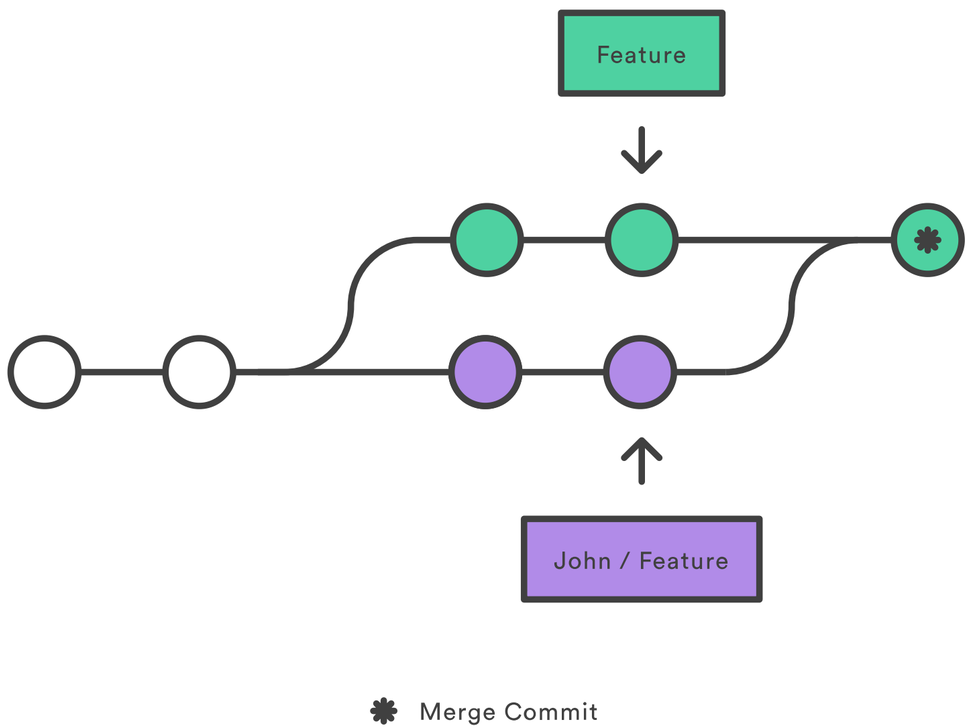
Versionamento de código - Git

- Rebase / Integração Linear -> segue a lógica do merge, porém apaga parte dos commits no histórico. Recomendado para ser usado entre branches de desenvolvedores, não diretamente da sua branch com a main, por exemplo.

Rebase/Integração linear



Merge/Fundir



Comandos do git

- Sintaxe de comandos e significados
  - git init -> avisa ao git para olhar para nossa pasta, cria um repositório git local
  - git add . ou git add nome-do-arquivo -> salva alterações em todos os arquivos da pasta ou em um arquivo especifico
  - git commit -m "mensagem do commit"
  - git status -> mostra o que foi modificado no nosso repo
  - git remote add origin url-do-repo ou ssh-do-repo -> adiciona um repositório remoto de origin do nosso repositório
  - git remote -v -> mostra qual repositório remoto está ligado ao nosso
  - git push origin nome-da-branch -> envia as alterações locais para repositório remoto
  - git clone url-do-repo -> faz uma cópia de repositório remoto no nosso computador
  - Branches:
    - git branch -> lista todas as branchs locais
    - git checkout -b nome-da-branch -> cria nova branch
    - git checkout nome-da-branch -> troca entre branch
  - git merge nome-da-branch -> puxa branch especificada e mergea com sua branch local

- Atualização entre Forks:
  - git clone url-do-seu-fork
  - git remote add upstream url-do-repo-original -> coloca url do repo original como o local do qual puxaremos atualizações para o nosso repo
  - git fetch (ou pull) upstream -> atualiza o repo local com o remoto original
  - git rebase (ou merge) upstream/main -> faz o merge das alterações do repo remoto na branch main com a sua branch

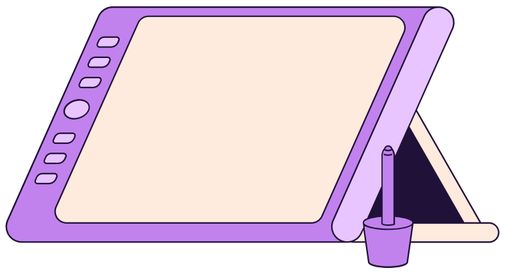
Configurando o Bash

- git config --global user.name "Tabita Barbosa" -> adiciona um nome da usuária git
- git config --global user.email “tabita.barbosa8@gmail.com” -> adiciona o email da usuária
- git config --list -> lista as usuárias no computador
- git config --global --unset user.name "Nome" -> retira o nome de uma usuária
- git config --global --unset user.email “nome@email.com” -> retira o email de uma usuária

Hora do Exercício

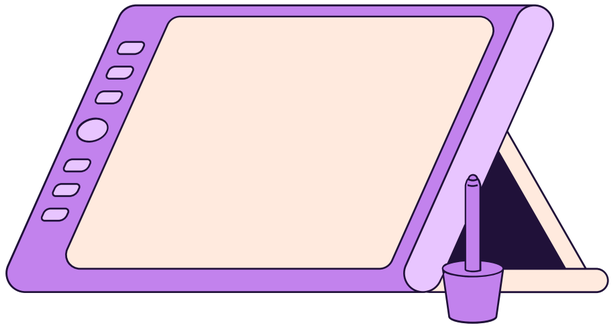
- Crie um repositório no GitHub, sem o Readme.md
- Crie uma pasta no seu computador
- Abra o git bash nela
- Adicione o seu repo remoto como o repo de origem
- Dê o comando git init
- Crie uma branch com seu nome
- Dentro da pasta crie um arquivo e escreva algo
- Adicione o arquivo ao git
- Faça um commit
- Envie as alterações para o seu repositório remoto

!! Atenção no último passo colocar o nome da sua branch local após o origin !!



Hora do Exercício

- Faça um fork do repositório -> github.com/tabita-barbosa/reprograma-s1
- A partir do seu fork, seu link, faça um clone
- Cria uma branch com seu nome
- Crie um arquivo com seu nome e escreva algo dentro
- Adicione esse arquivo ao olhar do git
- Faça um commit
- Faça um git push para o seu repositório
- No seu repositório no GitHub faça um Pull Request entre forks, ou seja envie a sua branch para a branch do repositório original



Fale comigo

Estou sempre aberta a perguntas e dúvidas!

E-mail

tabita.barbosa8@gmail.com

Linkedin

Tabita Barbosa

Telegram

@tabita_barbosa