Relatório 26 - Prática: Git e Github para Iniciantes (III)

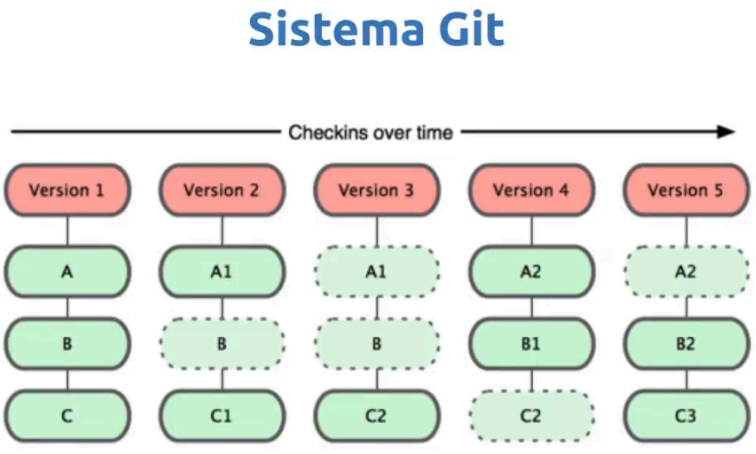
Lucas Augusto Nunes de Barros

Descrição das atividades

O Git foi criado por Linus Torvalds como uma resposta da necessidade de um sistema de controle de versão mais eficiente. A motivação surgiu quando o desenvolvedor do Linux enfrentou problemas com o sistema da *BitKeeper*, levando Torvalds a desenvolver o Git como uma alternativa ao sistema existente, e introduzindo ainda melhorias que o próprio Linus, como usuário do sistema *BitKeeper*, identificou em pontos falhos do então sistema de controle de versionamento.

Inicialmente foi projetado para gerenciar o desenvolvimento do kernel Linux, introduzindo um modelo que permite a cada desenvolvedor ter uma cópia completa do repositório. Manipulando as versões com operações rápidas e simples, mantendo uma robustez contra falhas, o Git evoluiu para se tornar o sistema de controle de versões mais utilizado por desenvolvedores ao redor do mundo.

Já o GitHub é uma plataforma web que utiliza o sistema de controle de versão Git e permite aos desenvolvedores armazenar, gerenciar e compartilhar projetos de código. O controle de versões com Git se baseia em registrar todas as alterações feitas nos arquivos de um repositório, criando um histórico completo que pode ser rastreado e revertido, se necessário.

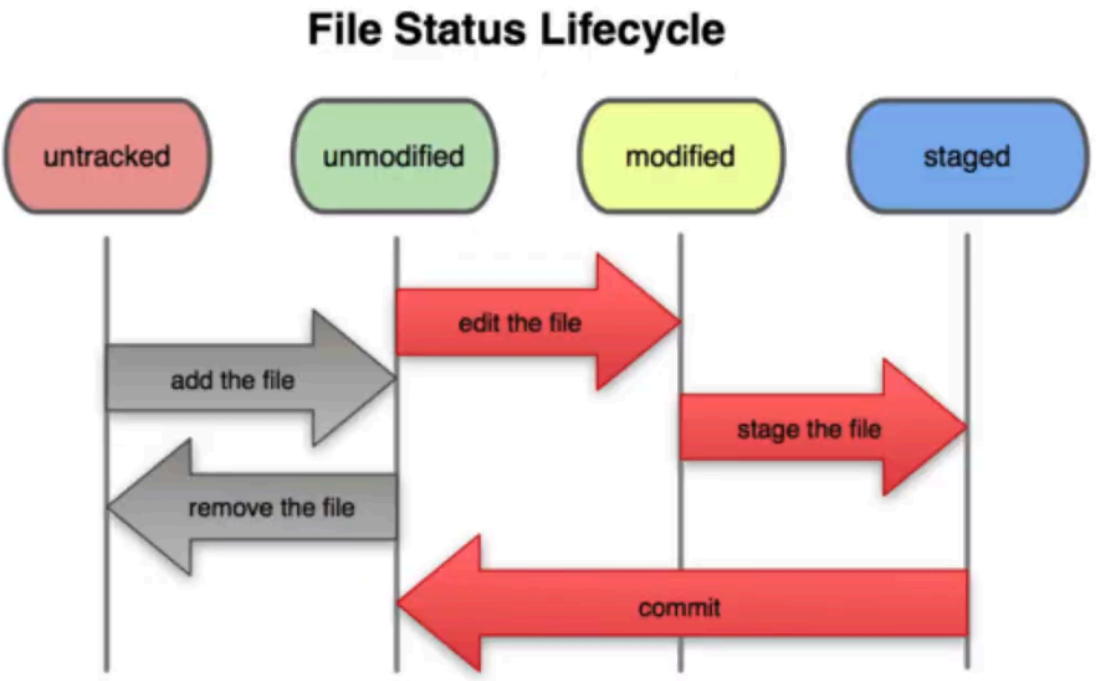


A colaboração em tempo real e a recuperação de versões anteriores são algumas vantagens que tornam o Git um sistema tão popular, ele aplica o conceito de *checkins over time* que faz referência ao acompanhamento contínuo de *commits* (algo como um *check-in*) ao longo do tempo, onde cada *commit* marca uma alteração específica no código, permitindo visualizar a evolução do projeto documento por documento, permitindo identificar alterações de forma pontual e seus respectivos autores.

Inicializando um repositório git

```
augusto@angband:~/Documentos$ cd ..
augusto@angband:~$ mkdir git-course
augusto@angband:~$ cd git-course/
augusto@angband:~/git-course$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Repositório vazio Git inicializado em  /home/augusto/git-course/.git/
augusto@angband:~/git-course$
```

É feita uma introdução ao editor de texto do terminal VIM, devido a prática como editor do terminal PICO…



```
augusto@angband:~/git-course$ git status
No ramo master

No commits yet

Arquivos não monitorados:
  (utilize "git add <arquivo>..." para incluir o que será submetido)
        Readme.md

nada adicionado ao envio mas arquivos não registrados estão presentes (use "git add" to registrar)
augusto@angband:~/git-course$
```

Git identificando o arquivo mas ainda sem “conhecê-lo”

```
augusto@angband:~/git-course$ git add Readme.md
augusto@angband:~/git-course$ git status
No ramo master

No commits yet

Mudanças a serem submetidas:
  (utilize "git rm --cached <arquivo>..." para não apresentar)
        new file:   Readme.md

augusto@angband:~/git-course$
```

Arquivos adicionados e prontos para serem *commitados.*

```
augusto@angband:~/git-course$ git commit -m "Add Readme.m"
[master (root-commit) 4e8a16c] Add Readme.m
 1 file changed, 1 insertion(+)
 create mode 100644 Readme.md
augusto@angband:~/git-course$
```

Commit realizado com sucesso. A partir dessa etapa os repositórios estarão sincronizados e o Git não reconhecerá mais o arquivo como modificado, pois o repositório já está atualizado. Ao alterar novamente algum arquivo contido no repositório local, novamente o Git identificará a mudança para então realizar um novo *commit.*

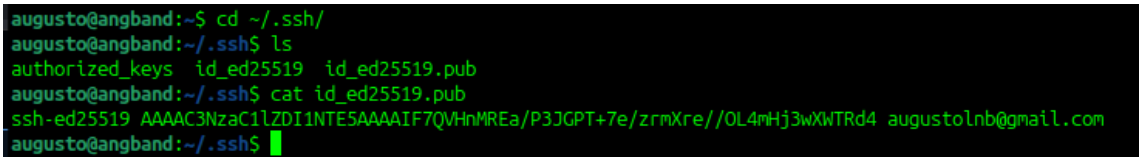
O log no Git é uma ferramenta útil para rastrear históricos de *commits*, permitindo visualizar alterações, o autor e a data em que tal alteração foi feita, proporcionando um acompanhamento detalhado da evolução do projeto. Torna-se particularmente útil para depuração, auditoria e desenvolvimento colaborativo, pois fornece uma visão clara das alterações no código ao longo do tempo.

```
augusto@angband:~/git-course$ git log
commit 4e8a16c2f8c66731cc216e176c784e4c1773cb7c (HEAD -> master)
Author: Lucas Augusto <augustolnb@gmail.com>
Date:   Fri Jun 6 13:21:44 2025 -0300

    Add Readme.m
augusto@angband:~/git-course$ git log --decorate
commit 4e8a16c2f8c66731cc216e176c784e4c1773cb7c (HEAD -> master)
Author: Lucas Augusto <augustolnb@gmail.com>
Date:   Fri Jun 6 13:21:44 2025 -0300

    Add Readme.m
augusto@angband:~/git-course$ git shortlog
Lucas Augusto (1):
      Add Readme.m

augusto@angband:~/git-course$
```

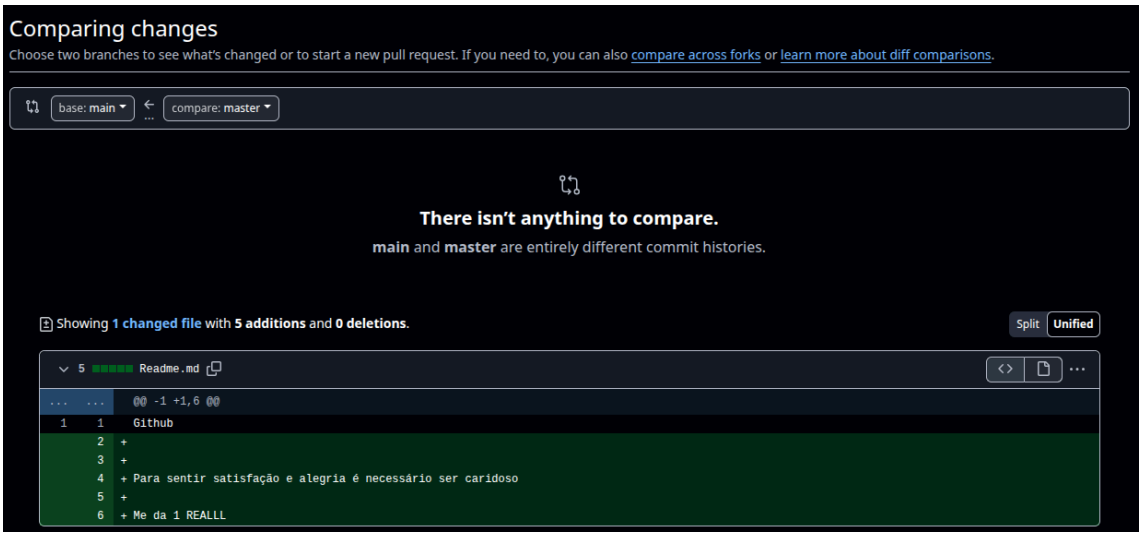
```
augusto@angband:~/git-course$ git diff
diff --git a/Readme.md b/Readme.md
index b4544e2..9fcd9a2 100644
--- a/Readme.md
+++ b/Readme.md
@@ -1 +1,6 @@
 Github
+
+
+Para sentir satisfação e alegria é necessário ser caridoso
+
+Me da 1 REALLL
augusto@angband:~/git-course$
```

Comandos	Descrição
git log	*Exibe o histórico completo de commits, em ordem cronológica, com detalhes como autor, data e mensagem.*
git shortlog	*Log reduzido, agrupa-se os commits por autor e listando suas contribuições.*
git show	*Mostra detalhes de um commit específico, incluindo alterações.*
git diff	*Compara alterações entre dois commits/branches quaisquer.*
git log --author=”nome”	*Busca apenas o log dos commits de um autor específico*
git log -p	*Mostra as diferenças detalhadas de cada commit.*
git log --name-only	*Exibe o histórico de commits incluindo apenas os nomes dos arquivos modificados.*
git reset	*Redefine o branch atual para um commit específico.*

As chaves SSH no Github são uma forma segura de autenticação para conectar um computador a um repositório remoto sem a necessidade de inserir usuário e senha repetidamente. Elas funcionam como uma identificação digital, composta por uma chave pública e uma chave privada . Ao ser corretamente configurada, a chave SSH permite ao desenvolvedor realizar operações no de repositórios com segurança. Esse mecanismo ajuda a garantir uma autenticação confiável.

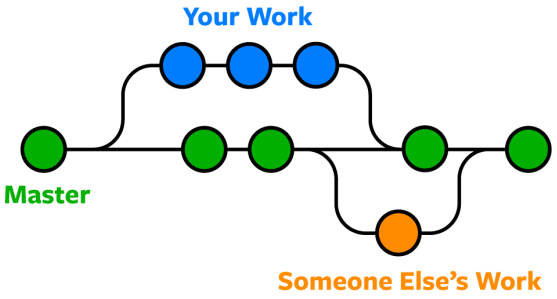


Além da configuração das chaves SSH é feita uma tentativa de atualizar o Github a partir dos comandos Git.



O *fork* no GitHub é um recurso que permite criar uma cópia de um projeto de outro usuário para o seu próprio perfil, possibilitando trabalhar no projeto sem afetar o original. Essa cópia independente pode ser modificada, sendo útil para contribuir com projetos de código aberto. Após realizar alterações, o desenvolvedor pode propor mudanças ao repositório original por meio de *pull requests*, facilitando a colaboração em projetos comunitários.

Já o *branch* no GitHub é um conceito fundamental que permite criar uma linha paralela de desenvolvimento dentro de um mesmo repositório, possibilitando trabalhar em novas funcionalidades, correções ou experimentos sem alterar o código principal (*branch master*). O *branch* é uma cópia independente do repositório em um determinado *commit*, onde alterações podem ser feitas e testadas isoladamente, evitando conflitos e riscos ao código original.

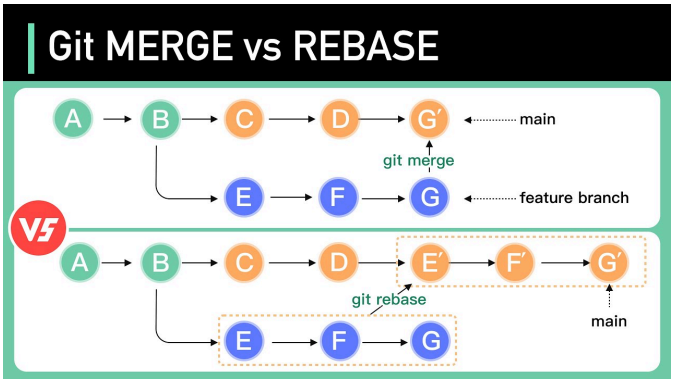


No GitHub, branches são usados para organizar fluxos de trabalho colaborativos e podem ser mesclados (merge) ao branch principal via pull requests, bem como facilitam a experimentação, pois podem ser criados, modificados ou excluídos sem alterar as demais partes do projeto.

git branch	Mostra os *branchs* existentes
git branch nome_do_branch	Cria o *branch*
git checkout nome_do_branch	Ativa o *branch*
git checkout -D nome_do_branch	Deleta o *branch*
git checkout -b nome_do_branch	Cria e ativa o *branch*

Os últimos métodos abordados são o *merge* e rebase, que no GitHub integram alterações de um branch a outro, mas com abordagens distintas. O *merge* é utilizado para unir históricos de dois *branches* distintos. Ele cria um novo *commit* que combina as alterações dos *branches* envolvidos. Esse método preserva o histórico completo, incluindo todas as ramificações e *commits* originais, o que facilita o rastreamento de alterações. É especialmente útil em projetos colaborativos, onde é preciso integrar as alterações de forma organizada.

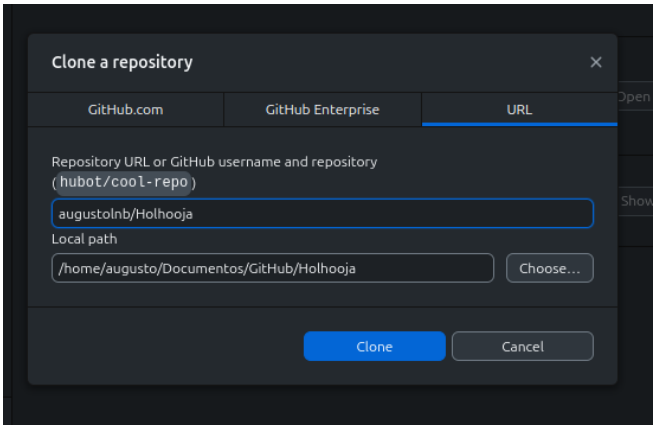
O *rebase*, por outro lado, reorganiza o histórico de *commits* ao mover a base de um *branch* para a ponta de outro. Em vez de criar um novo *commit*, são aplicados os *commits* do *branch* atual sobre o *branch* destino, reescrevendo-os, resultando em um histórico linear e sequencial. Essa abordagem elimina os *commits* redundantes do *merge* e simplifica a visualização do projeto, porém pode dificultar o rastreamento de alterações em ambientes colaborativos, pois altera o histórico existente.

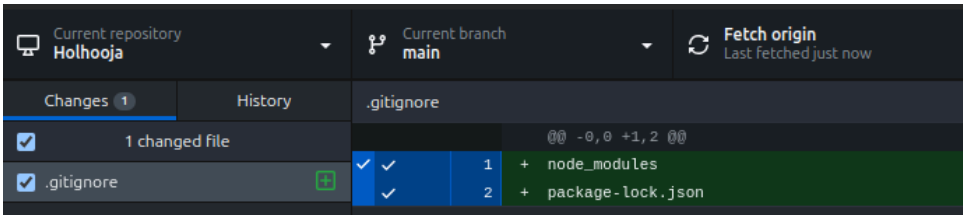


A principal diferença entre *merge* e *rebase* está na forma como lidam com o histórico.

2. GitHub Desktop

O GitHub Desktop é uma aplicação com interface gráfica que facilita a interação com repositórios no GitHub, sendo ideal para usuários que preferem evitar comandos de terminal e gostam de interfaces mais amigáveis. Ele permite clonar repositórios, criar *branches*, realizar *commits*, e enviar atualizações (*push)* ou ainda buscar atualizações (*pull)* de forma simplificada, oferecendo um histórico mais visual das alterações realizadas. Abaixo algumas alterações feitas com o GitHub Desktop.





Conclusão

Em conclusão, o Git e o GitHub oferecem um conjunto poderoso de ferramentas para controle de versão e trabalhos colaborativos de desenvolvimento de software. Seus recursos permitem fluxos de trabalho eficientes e organizados. No GitHub, a integração facilita a revisão e a colaboração, tornando o desenvolvimento mais ágil e confiável, tanto para projetos individuais quanto para equipes.

Referências

[1] GIT-SCM.COM. git-log Documentation.

Disponível em: <https://git-scm.com/docs/git-log>.

Acessado em 09 de junho de 2025

[2] GEEKSFORGEEKS.ORG. What's The Difference Between git reset –mixed, –soft, and –hard? | GeeksforGeeks.

Disponível em: <https://www.geeksforgeeks.org/whats-the-difference-between-git-reset-mixed-soft-and-hard/>.

Acessado em 09 de junho de 2025

[3] DOCS.GITHUB.COM. Adding locally hosted code to GitHub.

Disponível em: <https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github>.

Acessado em 10 de junho de 2025

[4] NOBLEDESKTOP.COM. Git Branches: List, Create, Switch to, Merge, Push, & Delete

Disponível em: <https://www.nobledesktop.com/learn/git/git-branches>

Acessado em 10 de junho de 2025