Atividade: Controle de Versão com Git e GitHub

Objetivo

Nesta atividade, você irá trabalhar com os conceitos básicos do Git e GitHub, incluindo:

- Criar um repositório local e versionar arquivos com Git.
- Conectar um **repositório local** a um **repositório remoto** no GitHub.
- Utilizar commits, branches e pull requests para simular um fluxo real de trabalho.
- Elaborar um relatório explicativo sobre as funcionalidades do Git utilizadas na atividade.

Instruções

Criando o Repositório Local

1. Crie uma pasta chamada projeto-git no seu computador.
2. No terminal, entre na pasta e inicie um repositório Git:

```
git init
```

3. Crie um arquivo README.md e adicione uma breve descrição do projeto.
4. Adicione o arquivo ao versionamento e faça um commit:

```
git add README.md
git commit -m "Adiciona README inicial"
```

Criando o Repositório no GitHub e Conectando ao Local

1. Acesse o GitHub e crie um novo repositório sem README.
2. No terminal, conecte o repositório remoto:

```
git remote add origin https://github.com/usuario/projeto-git.git
```

3. Envie o commit inicial para o GitHub: git push -u origin main

Criando e Trabalhando com Branches

1. Crie uma nova branch para adicionar funcionalidades:

```
git checkout -b feature-nova
```

2. Crie um arquivo index.html com um código básico de HTML.
3. Adicione e faça um commit das alterações:

```
git add index.html
git commit -m "Cria página inicial"
```

Enviando Alterações e Criando um Pull Request

1. Envie a branch para o GitHub:

```
git push origin feature-nova
```

2. No GitHub, abra um Pull Request da branch feature-nova para main.
3. Peça para um colega revisar seu código antes de aceitar o merge.
4. Após a aprovação, faça o merge e delete a branch feature-nova.

Baixando Atualizações e Finalizando

1. No terminal, volte para a branch main:
git checkout main
2. Baixe as atualizações do repositório remoto:
git pull origin main
3. Confirme que os arquivos da branch feature-nova agora estão na main.

Entrega do Relatório

- Após concluir a atividade, cada aluno deverá entregar um relatório contendo:
- Uma descrição de cada funcionalidade do Git utilizada na atividade (exemplo: git init, git commit, git push, git pull, git checkout, etc).
- O que **cada comando faz** e sua **importância** no controle de versão.
- Screenshots ou trechos de código comprovando a execução dos passos.
- Uma reflexão sobre a importância do Git no desenvolvimento de software.

Formato: PDF

Critérios de Avaliação
Criou e configurou corretamente o repositório local e remoto?
Utilizou commits organizados e com mensagens claras?
Criou e gerenciou branches corretamente?
Criou um pull request e interagiu no GitHub?
Elaborou um relatório bem estruturado, explicando os conceitos utilizados?
Prazo de Entrega: 07/03/2025
Forma de Entrega: Subir o relatório no repositório e enviar o link no AVA

Relatório de Controle de Versão com Git e GitHub

1. Introdução

Neste projeto, desenvolvi um **site de entretenimento** que exibe uma lista de recomendações de **filmes**, **séries** e **livros**. O objetivo do projeto foi não apenas criar um site visualmente atraente, mas também aprender a utilizar **Git** e **GitHub** para controle de versão, colaboração e gerenciamento de código.

Objetivo do Site:

- Criar uma página HTML simples que apresenta uma lista de filmes, séries e livros recomendados.
- Utilizar **Git** para versionar e organizar o código fonte.
- Subir o projeto para o **GitHub** e gerenciar as alterações com **commits**, **branches** e **pull requests**.

2. Funcionalidades do Git Utilizadas

Durante a implementação do projeto, utilizei várias funcionalidades do **Git** para gerenciar as alterações e controlar o versionamento do código. A seguir, descrevo as funcionalidades do Git utilizadas e sua importância:

2.1. git init

O comando `git init` foi utilizado para **inicializar** um repositório Git local. Esse comando cria um diretório `.git` na pasta do projeto, permitindo que o Git comece a rastrear alterações nos arquivos. Esse passo é essencial para iniciar o controle de versão.

Exemplo: git init

2.2. git add

O comando `git add` é usado para adicionar alterações no código à **área de preparação (staging area)**. Este comando prepara os arquivos para serem registrados no commit. A importância de usar esse comando é garantir que apenas as alterações desejadas sejam incluídas no próximo commit.

Exemplo: git add README.md

2.3. git commit

O comando `git commit` registra as alterações feitas no repositório local. Ao usar `git commit`, a mudança se torna parte do histórico do repositório, com uma mensagem explicativa. É fundamental para manter o controle sobre as versões e facilitar a colaboração com outros desenvolvedores.

Exemplo: git commit -m "Adiciona README inicial sobre entretenimento"

2.4. git remote add

O comando `git remote add` é usado para **conectar** o repositório local com um repositório remoto no GitHub. Isso é crucial para garantir que o código possa ser compartilhado e sincronizado com outros desenvolvedores.

Exemplo: git remote add origin https://github.com/usuario/projeto-git.git

2.5. git push

O comando `git push` é usado para **enviar** as alterações feitas no repositório local para o repositório remoto no GitHub. Esse comando permite que as alterações feitas no código sejam visíveis para outros colaboradores e também mantém o repositório remoto atualizado.

Exemplo: git push -u origin main

2.6. git checkout

O comando `git checkout` é usado para **alternar entre diferentes branches**. Ao trabalhar com diferentes funcionalidades ou recursos, criar uma nova branch permite que você trabalhe em uma versão isolada do projeto, sem afetar a branch principal.

Exemplo: git checkout -b feature-nova

2.7. git pull

O comando `git pull` é usado para **baixar as atualizações** do repositório remoto e integrá-las ao repositório local. Esse comando é essencial para garantir que você tenha a versão mais atualizada do código antes de fazer novas alterações.

Exemplo: git pull origin main

2.8. git merge

O comando `git merge` é utilizado para **mesclar** uma branch com outra. Neste projeto, usei esse comando para mesclar a branch `feature-nova` na branch `main` após concluir as modificações na página inicial.

Exemplo: git merge feature-nova

2.9. Pull Request no GitHub

No GitHub, criei um **Pull Request** para revisar e mesclar a branch `feature-nova` com a branch `main`. O Pull Request facilita a colaboração entre desenvolvedores, permitindo que as alterações sejam revisadas antes de serem mescladas ao código principal.

3. Como o Git foi utilizado no desenvolvimento do site

3.1. Criando o Repositório Local e Remoto

O primeiro passo foi criar o repositório Git no meu computador com o comando `git init`, seguido de um commit inicial com o arquivo `README.md` para descrever o projeto. Em seguida, criei um repositório remoto no GitHub e conectei-o ao meu repositório local utilizando o comando `git remote add`.

3.2. Desenvolvimento do Site

Durante o desenvolvimento, criei uma estrutura HTML simples com listas de **filmes**, **séries** e **livros** recomendados, e também adicionei um arquivo CSS para estilizar o layout. As listas de entretenimento foram organizadas de forma visualmente agradável, com efeitos de hover para interação.

3.3. Gerenciando Alterações com Git

Conforme fui desenvolvendo o site, utilizei o Git para versionar minhas alterações. Cada funcionalidade e modificação importante foi registrada em um **commit**, com mensagens claras e descritivas. Criei uma branch chamada `feature-nova` para adicionar as listas de filmes, séries e livros, e depois mesclei essa branch com a `main` usando o comando `git merge`.

3.4. Interação no GitHub

Enviei minhas alterações para o repositório remoto no GitHub usando o comando `git push`. Criei um **Pull Request** no GitHub para revisar as alterações na branch `feature-nova`, realizar a aprovação e mesclar as alterações na branch principal.

4. Screenshots e Trechos de Código

Criando o Repositório Local

1. Crie uma pasta chamada projeto-git no seu computador.

2. No terminal, entre na pasta e inicie um repositório Git:

git init

3. Crie um arquivo README.md e adicione uma breve descrição do projeto.

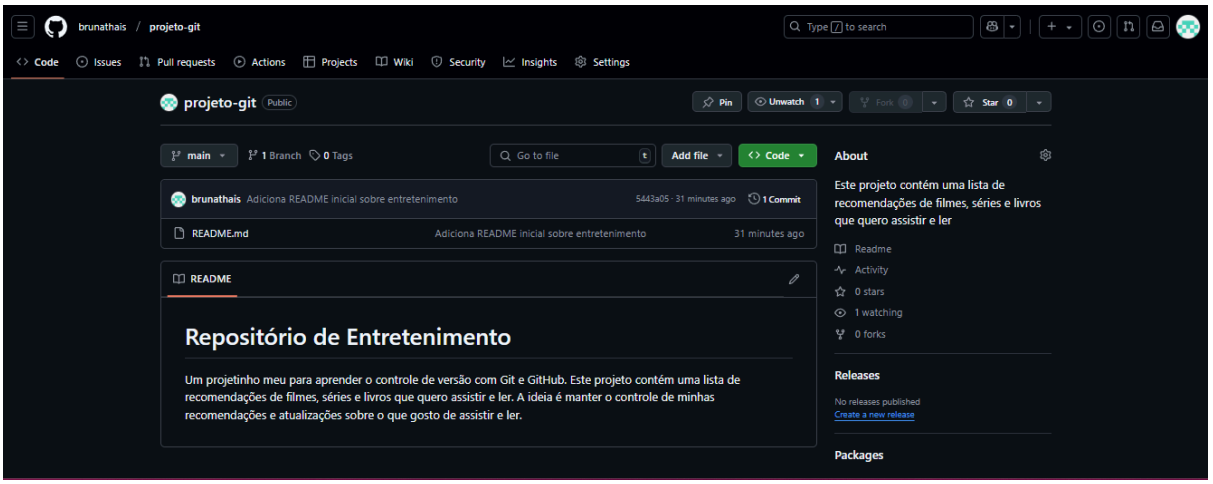
4. Adicione o arquivo ao versionamento e faça um commit:

git add README.md

git commit -m "Adiciona README inicial"

```
PS C:\Projeto-git> git init
Initialized empty Git repository in C:/Projeto-git/.git/
PS C:\Projeto-git> git add README.md
PS C:\Projeto-git> git commit -m "Adiciona README inicial"
[master (root-commit) 1ba9d27] Adiciona README inicial
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

Criando o Repositório no GitHub e Conectando ao Local



1. Acesse GitHub e crie um novo repositório sem README.

2. No terminal, conecte o repositório remoto:

git remote add origin https://github.com/usuario/projeto-git.git

3. Envie o commit inicial para o GitHub:

git push -u origin main

```
PS C:\Projeto-git> git remote add origin https://github.com/1Debis1/Projeto-git.git
PS C:\Projeto-git> git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/1Debis1/Projeto-git.git'
PS C:\Projeto-git> git branch
* master
PS C:\Projeto-git> git branch -m master main
PS C:\Projeto-git> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 20 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 450 bytes | 450.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/1Debis1/Projeto-git.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Criando e Trabalhando com Branches

```
PS C:\Projeto-git> git checkout -b feature-nova
Switched to a new branch 'feature-nova'
PS C:\Projeto-git> git commit -m "Entreterimento!"
On branch feature-nova
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Index.html
        main.css

no changes added to commit (use "git add" and/or "git commit -a")
```

```
PS C:\Projeto-git> git add .
PS C:\Projeto-git> git commit -m "Entreterimento!"
[feature-nova 097adc2] Entreterimento!
 3 files changed, 210 insertions(+)
 create mode 100644 Index.html
 create mode 100644 main.css
```

1. Crie uma nova branch para adicionar funcionalidades:

git checkout -b feature-nova

2. Crie um arquivo index.html com um código básico de HTML.

3. Adicione e faça um commit das alterações:

git add index.html

git commit -m "Cria página inicial"

Enviando Alterações e Criando um Pull Request

```
PS C:\Projeto-git> git push origin feature-nova
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.37 KiB | 607.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-nova' on GitHub by visiting:
remote:      https://github.com/1Debis1/Projeto-git/pull/new/feature-nova
remote:
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
 * [new branch]      feature-nova -> feature-nova
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
 * [new branch]      feature-nova -> feature-nova
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
 * [new branch]      feature-nova -> feature-nova
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
To https://github.com/1Debis1/Projeto-git.git
 * [new branch]      feature-nova -> feature-nova
```

1. Envie a branch para o GitHub:

git push origin feature-nova

2. No GitHub, abra um Pull Request da branch feature-nova para main.

3. Peça para um colega revisar seu código antes de aceitar o merge.

4. Após a aprovação, faça o merge e delete a branch feature-nova.

Baixando Atualizações e Finalizando

```
PS C:\Projeto-git> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Projeto-git> git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (1/1), 914 bytes | 457.00 KiB/s, done.
From https://github.com/1Debis1/Projeto-git
 * branch            main       -> FETCH_HEAD
   1ba9d27..f7d5477  main       -> origin/main
Updating 1ba9d27..f7d5477
Fast-forward
 Index.html | 114 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 README.md  |   3 ++
 main.css   |  93 +++++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 210 insertions(+)
 create mode 100644 Index.html
 create mode 100644 main.css
PS C:\Projeto-git>
```

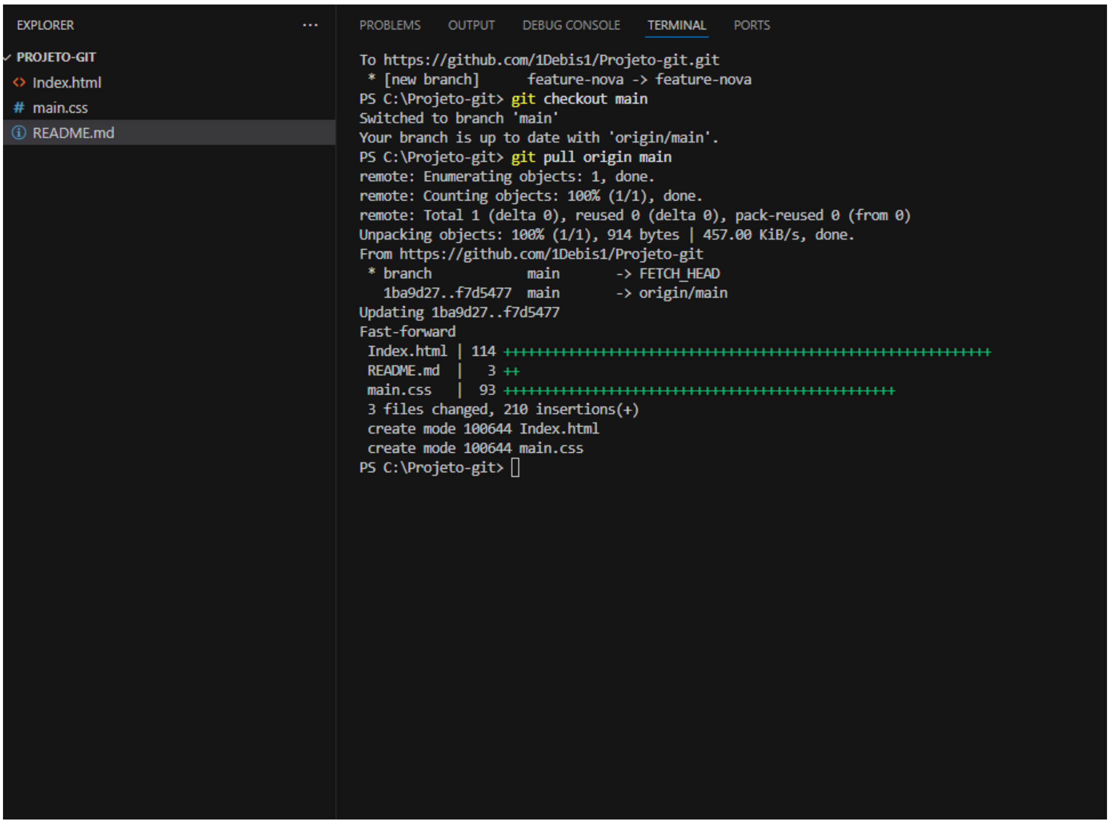
1. No terminal, volte para a branch main:

git checkout main

2. Baixe as atualizações do repositório remoto:

git pull origin main

3. Confirme que os arquivos da branch feature-nova agora estão na main.



Exemplo de código HTML:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Entretenimento | Git</title>
    <link rel="stylesheet" href="main.css">
</head>
<body>
    <header>
        <h1>Entretenimento! :D</h1>
        <p>Um projetinho meu para aprender o controle de versão com Git e GitHub</p>
    </header>
    <main>
        <h2>Lista de Filmes, Séries e Livros</h2>
        <!-- Conteúdo do site -->
    </main>
    <footer>
        <p>Projeto de Entretenimento - 2025</p>
    </footer>
</body>
</html>
```

5. Reflexão sobre a Importância do Git no Desenvolvimento de Software

O **Git** é uma ferramenta fundamental para o desenvolvimento de software, pois permite que o código seja versionado, facilitando o controle de alterações, a colaboração entre equipes e o gerenciamento de diferentes versões de um projeto. Ao usar o Git, podemos realizar **commits** organizados, controlar as alterações através de **branches**, e revisar o código com **Pull Requests**.

Além disso, o **GitHub** proporciona uma plataforma de colaboração remota, onde as equipes podem trabalhar simultaneamente no mesmo projeto, sem se preocupar com conflitos de código. O uso de **branches** permite que novas funcionalidades ou correções sejam desenvolvidas de forma isolada, sem afetar a versão principal do projeto.

A integração do Git com plataformas como GitHub transforma o desenvolvimento de software em um processo mais eficiente, organizado e colaborativo, ajudando a manter a qualidade do código ao longo do tempo.

Conclusão

Neste projeto, aprendi a utilizar as funcionalidades básicas do Git e GitHub, que são essenciais para o controle de versão e colaboração no desenvolvimento de software. A prática de realizar **commits**, gerenciar **branches** e interagir com **Pull Requests** no GitHub me ajudou a entender melhor como essas ferramentas podem otimizar o fluxo de trabalho em projetos de desenvolvimento.