

UNIVERSIDADE FEDERAL DE LAVRAS – UFLA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Gustavo Soares Silva
202120103

RELATÓRIO DO PROJETO PRÁTICO
DE GESTÃO DE VERSÕES
ENGENHARIA DE SOFTWARE

MINISTRADA PELO DOCENTE
ANTÔNIO MARIA PEREIRA DE RESENDE

LAVRAS - MG

2023

Seção Criação de Projeto e Clonagem	3
Criação do Projeto no GitHub (browser) e clonagem	3
Criação das duas classes no Visual Studio Code	3
Seção alteração e atualização do projeto	4
Edição da classe A e criação da classe C	4
Status do Repositório	4
Commit das modificações nas classes A e C	5
Seção de rotulação (tag) de versões liberadas (releases) do projeto e uso de tags	5
Criação da Tag “Versão 1.0” no último commit	5
Criação da Release “Release Versão 1.0” no GitHub (browser)	6
Criação da classe D e modificações nas classes B e C, commit dessas modificações e criação da Tag “Versão 2.0”	6
Criação da Release “Release Versão 2.0”	7
Download da Release “Release Versão 1.0”	7
Seção de resolução de conflitos (merge)	8
Edição da classe D localmente	8
Edição da classe D pelo GitHub (browser)	8
Commit dos dois arquivos e push do arquivo local	9
Resultado final do Merge da classe D no GitHub (browser)	9
Seção de verificar diferenças entre um mesmo arquivo em versões diferentes e diferença entre versões	10
Diferenças na classe B entre as versões 1.0 e 2.0	10
Diferenças na classe D entre as versões 1.0 e 2.0	10
Diferenças entre as versões 1.0 e 2.0	11
Outras ferramentas e relatórios	12
Perguntas	12
A evolução do meu projeto	13


Seção Criação de Projeto e Clonagem

Criação do Projeto no GitHub (browser) e clonagem

Create a new repository

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

Owner *

 GustavoSilva36

 /

Repository name *

Projeto_Gustavo_Soares

✔ Projeto_Gustavo_Soares is available.

Great repository names are short and memorable. Need inspiration? How about [cautious-train?](#)

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:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Criação das duas classes no Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
ClasseA.cpp - Projeto_Gustavo_Soares - Visual Studio Code

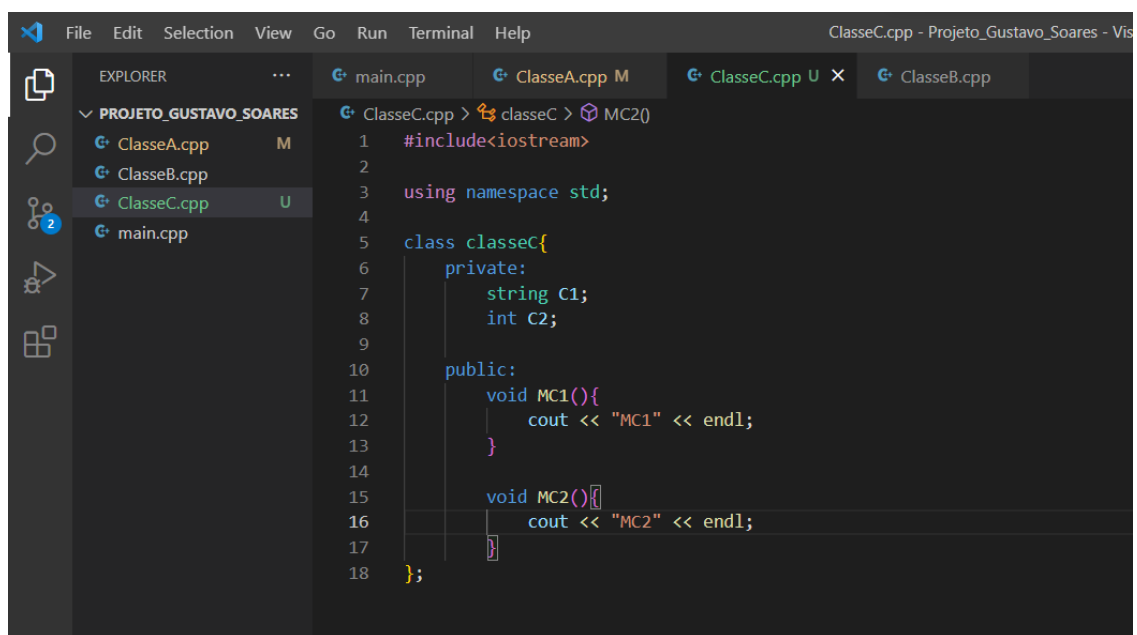
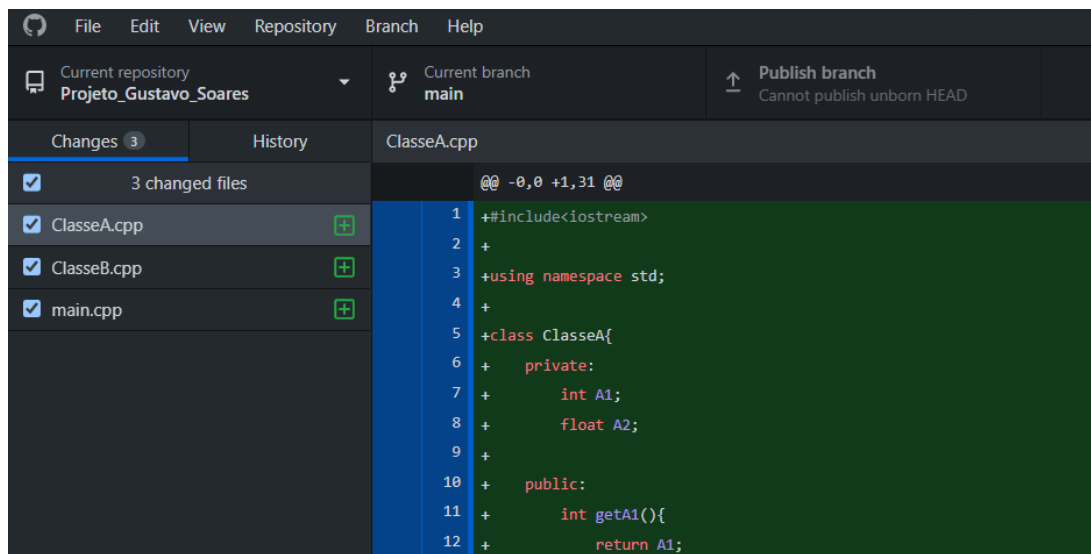
EXPLORER
PROJETO_GUSTAVO_SOARES
  ClasseA.cpp
  ClasseB.cpp
  main.cpp

ClasseA.cpp
1 #include<iostream>
2
3 using namespace std;
4
5 class ClasseA{
6     private:
7         int A1;
8         float A2;
9
10    public:
11        int getA1(){
12            return A1;
13        }
14        void setA1(int A1){
15            this->A1 = A1;
16        }
17        float getA2(){
18            return A2;
19        }
20        void setA2(float A2){
21            this->A2 = A2;
22        }
23
24        void MA1(){
25            cout << "MA1" << endl;
26        }
27
28        void MA2(){
29            cout << "MA2" << endl;
30        }
31    };

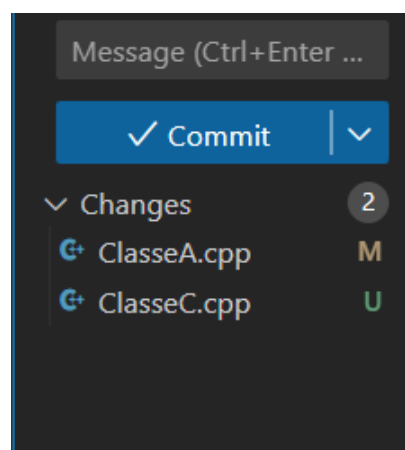
ClasseB.cpp
1 #include<iostream>
2
3 using namespace std;
4
5 class classeB{
6     private:
7         int B1;
8         float B2;
9
10    public:
11        int getB1(){
12            return B1;
13        }
14        void setB1(int B1){
15            this->B1 = B1;
16        }
17        float getB2(){
18            return B2;
19        }
20        void setB2(float B2){
21            this->B2 = B2;
22        }
23
24        void MB1(){
25            cout << "MB1" << endl;
26        }
27
28        void MB2(){
29            cout << "MB2" << endl;
30        }
31    };
```

Seção alteração e atualização do projeto

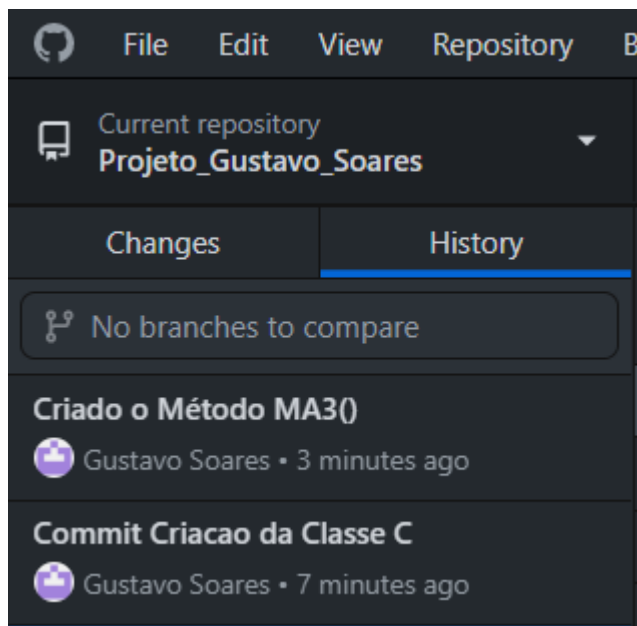
Edição da classe A e criação da classe C



Status do Repositório

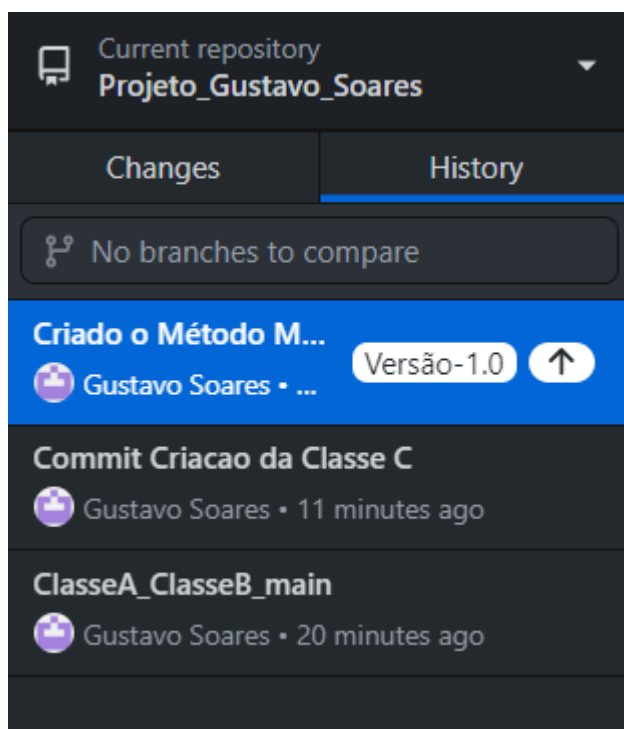


Commit das modificações nas classes A e C

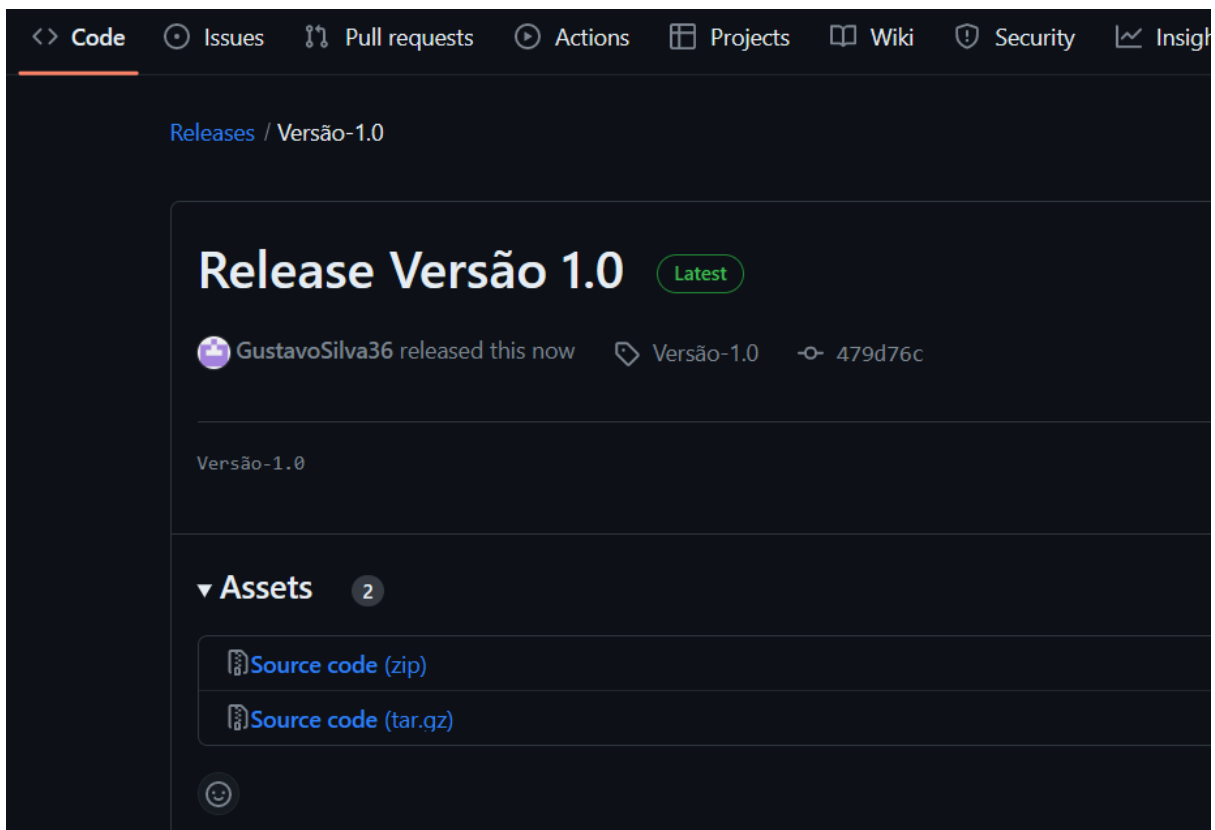


Seção de rotulação (tag) de versões liberadas (releases) do projeto e uso de tags

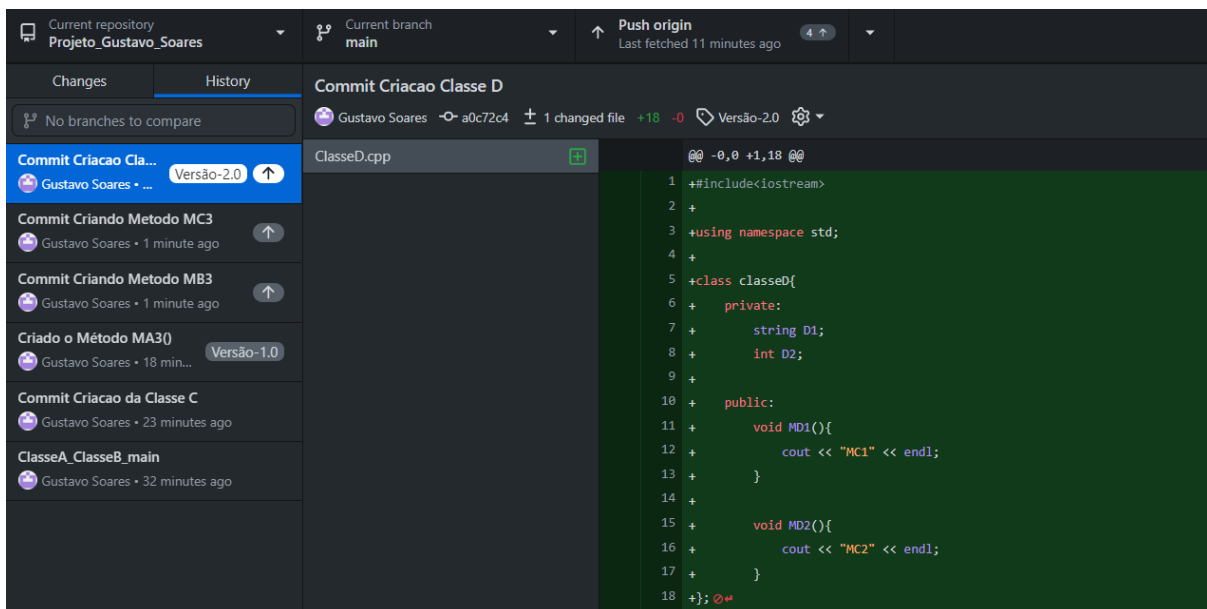
Criação da Tag “Versão 1.0” no último commit



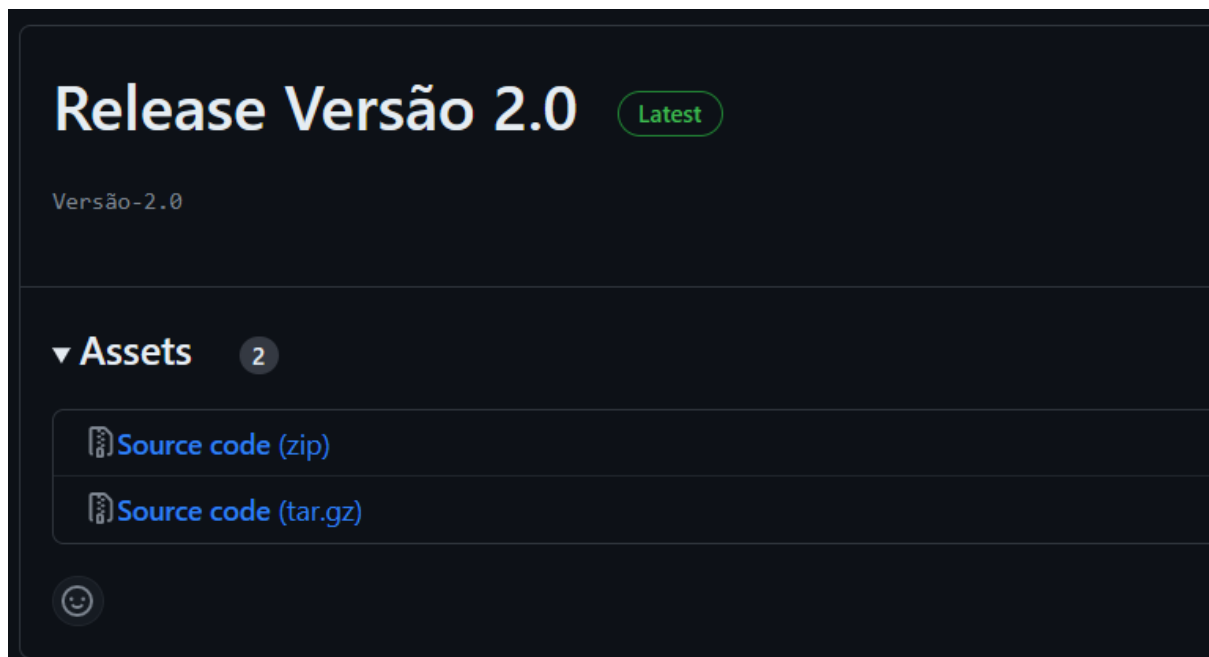
Criação da Release “Release Versão 1.0” no GitHub (browser)



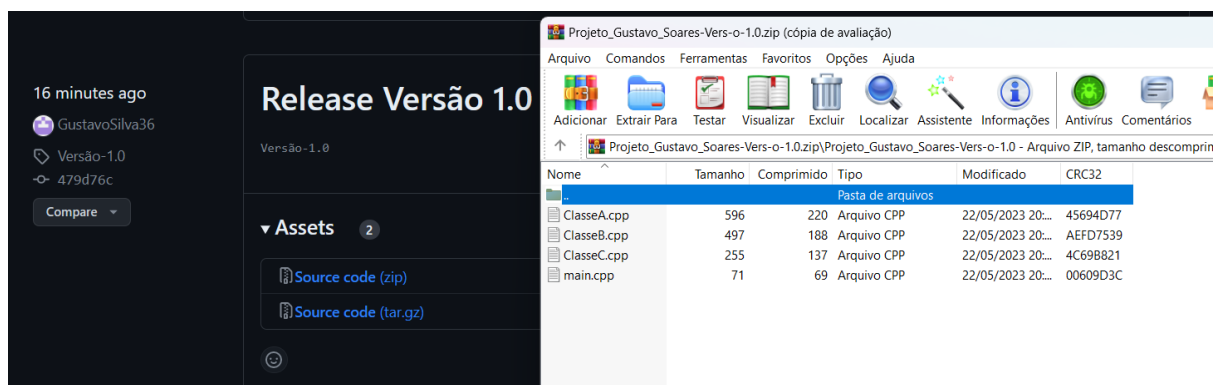
Criação da classe D e modificações nas classes B e C, commit dessas modificações e criação da Tag “Versão 2.0”



Criação da Release “Release Versão 2.0”



Download da Release “Release Versão 1.0”



Não foi possível fazer um Clone da “Versão 1.0”, mas foi possível fazer o download com .zip como mostrado no print acima.

Seção de resolução de conflitos (merge)

Edição da classe D localmente

```
using namespace std;

class classeD{
private:
    string D1;
    int D2;

public:
    void MD1(){
        cout << "MD1" << endl;
    }

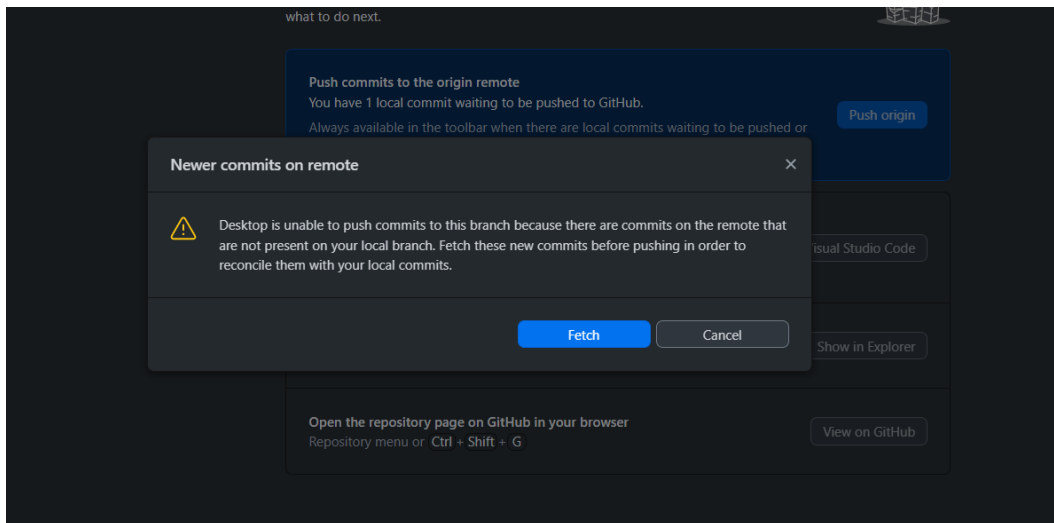
    void MD2(){
        cout << "MD2" << endl;
    }

    void MD3(){
        cout << "MD3" << endl;
    }
};
```

Edição da classe D pelo GitHub (browser)

```
4
5     class classeD{
6         private:
7             string D1;
8             int D2;
9
10        public:
11            void MD1(){
12                cout << "MD1" << endl;
13            }
14
15            void MD2(){
16                cout << "MD2" << endl;
17            }
18
19            void MD4(){
20                cout << "MD4" << endl;
21            }
22        };
23
```


Commit dos dois arquivos e push do arquivo local



Essa foi a mensagem que apareceu quando foi feito o push. Houve conflito entre os arquivos, dessa forma foi necessário fazer um Merge.

Resultado final do Merge da classe D no GitHub (browser)

```
GustavoSilva36 Merge branch 'main' of https://github.com/GustavoSilva36/Projeto_Gust... a397387 · 6 minutes ago History
Code Blame 26 lines (20 loc) · 392 Bytes Raw Copy Download Edit History
1 #include<iostream>
2
3 using namespace std;
4
5 class classeD{
6     private:
7         string D1;
8         int D2;
9
10    public:
11        void MD1(){
12            cout << "MD1" << endl;
13        }
14
15        void MD2(){
16            cout << "MD2" << endl;
17        }
18
19        void MD3(){
20            cout << "MD3" << endl;
21        }
22
23        void MD4(){
24            cout << "MD4" << endl;
25        }
26    };
```

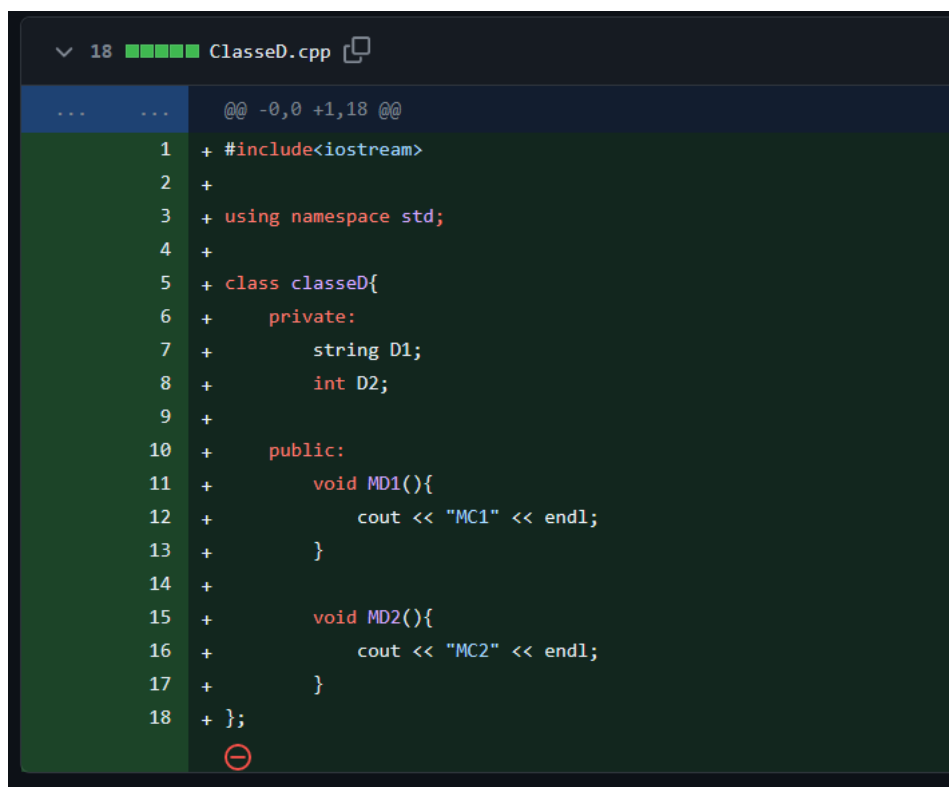
Seção de verificar diferenças entre um mesmo arquivo em versões diferentes e diferença entre versões

Diferenças na classe B entre as versões 1.0 e 2.0



```
@@ -28,4 +28,8 @@ class classeB{
28      28      void MB2(){
29      29          cout << "MB2" << endl;
30      30      }
31      31      +
32      32      + void MB3(){
33      33          cout << "MB3" << endl;
34      34      +
31      35      };
-
```

Diferenças na classe D entre as versões 1.0 e 2.0



```
@@ -0,0 +1,18 @@
1 + #include<iostream>
2 +
3 + using namespace std;
4 +
5 + class classeD{
6 +     private:
7 +         string D1;
8 +         int D2;
9 +
10 +     public:
11 +         void MD1(){
12 +             cout << "MC1" << endl;
13 +         }
14 +
15 +         void MD2(){
16 +             cout << "MC2" << endl;
17 +         }
18 + };
-
```

Diferenças entre as versões 1.0 e 2.0

Showing 3 changed files with 26 additions and 0 deletions.

4 ClasseB.cpp

```
@@ -28,4 +28,8 @@ class classeB{
28      28          void MB2(){
29      29              cout << "MB2" << endl;
30      30          }
31      31      +
32      32          void MB3(){
33      33              cout << "MB3" << endl;
34      34          }
31      35      };
      35      -
```

4 ClasseC.cpp

```
@@ -15,4 +15,8 @@ class classeC{
15      15          void MC2(){
16      16              cout << "MC2" << endl;
17      17          }
18      18      +
19      19          void MC3(){
20      20              cout << "MC3" << endl;
21      21          }
18      22      };
      22      -
```

18 ClasseD.cpp

```
@@ -0,0 +1,18 @@
...      ...
1      + #include<iostream>
2      +
3      + using namespace std;
4      +
5      + class classeD{
6      +     private:
7      +         string D1;
8      +         int D2;
9      +
10     +     public:
11     +         void MD1(){
12     +             cout << "MC1" << endl;
13     +         }
14     +
15     +         void MD2(){
16     +             cout << "MC2" << endl;
17     +         }

```

Outras ferramentas e relatórios

Perguntas

1. A ferramenta disponibiliza um relatório que mostra todas as alterações já feitas no repositório?

Sim, o GitHub disponibiliza uma ferramenta chamada "Insights" (Análises, em português) que oferece várias informações e estatísticas sobre um repositório, incluindo um histórico de alterações. Essa funcionalidade permite que você visualize as atividades do repositório ao longo do tempo, como commits, pull requests, issues fechadas, entre outros.

2. A ferramenta disponibiliza um relatório que mostra todas as versões/alterações de um único componente?

O GitHub em si não possui uma ferramenta específica para rastrear as versões ou alterações de um único componente dentro de um repositório. No entanto, é possível utilizar recursos do Git, que é o sistema de controle de versão utilizado pelo GitHub, para obter informações sobre as alterações em um componente específico.

3. A ferramenta disponibiliza um relatório contendo as diferenças entre as versões?

Sim, o GitHub fornece uma funcionalidade que permite visualizar as diferenças entre as versões de um arquivo. Isso é feito por meio do recurso de "Comparação" (ou "Compare") do GitHub.

4. Quais são os outros relatórios que a ferramenta disponibiliza?

O Git é uma ferramenta de controle de versão distribuída que se concentra principalmente em rastrear alterações em um repositório de código-fonte. Embora não forneça relatórios prontos para uso, o Git oferece uma variedade de recursos e comandos que podem ser usados para extrair informações úteis e gerar relatórios personalizados. Alguns exemplos de relatórios que podem ser obtidos utilizando recursos do Git incluem:

- Histórico de commits: O comando "git log" permite visualizar o histórico completo de commits em um repositório, incluindo detalhes como o autor, a data e a mensagem associada a cada commit.
- Estatísticas de contribuição: O Git oferece recursos para rastrear a atividade de contribuição de cada autor no projeto.
- Ramificações (branches) e mesclagens (merges): O Git permite visualizar informações sobre as ramificações criadas no repositório e as mesclagens realizadas entre elas.
- Diferenças entre tags: As tags no Git são marcadores que podem ser usados para marcar versões específicas do código.

5. A ferramenta disponibiliza uma interface gráfica para gerenciar a árvore de versões?

Sim, o GitHub oferece uma interface gráfica para gerenciar a árvore de versões de um repositório. Essa interface gráfica é fornecida por meio da página "Branches" (Ramos) do repositório, onde é possível visualizar, criar, mesclar e excluir ramos (branches) e tags.

A evolução do meu projeto

FileEditViewRepositoryBranchHelp

Current repository

Projeto_Gustavo_Soares

Current branch

main

Fetch origin

Last fetched 13 minutes ago

Changes

History

No branches to compare

Merge branch 'main' of https://github.com/GustavoSilva36/Projeto_Gustavo...

Gustavo Soares

a397387

1 changed file

+5-1

... Soares

ClasseD.cpp

1919

2020

2121

22

22

23

24

25

26

@@ -19,4 +19,8 @@ class classeD{

void MD3(){

cout << "MD3" << endl;

}

-};

+void MD4(){

+cout << "MD4" << endl;

+}

+};

Update ClasseD.cpp Metodo MD3

Gustavo Soares • 2 hours ago

Update ClasseD.cpp Metodo MD4

GustavoSilva36 • 2 hours ago

Commit Criacao Classe D

Gustavo Soares • 2 hour...

Versão-2.0

Commit Criando Metodo MC3

Gustavo Soares • 2 hours ago

Commit Criando Metodo MB3

Gustavo Soares • 2 hours ago

Criado o Método MA3()

Gustavo Soares • 2 hour...

Versão-1.0

Commit Criacao da Classe C

Gustavo Soares • 3 hours ago

ClasseA_ClasseB_main

Gustavo Soares • 3 hours ago