

**FIAP** GRADUAÇÃO

**DISCIPLINA: PROJETO DE SISTEMAS APLICADO AS MELHORES PRÁTICAS EM  
QUALIDADE DE SOFTWARE E GOVERNANÇA DE TI**

**AULA:**

**1 – INTRODUÇÃO ÀS BOAS PRÁTICAS DE DESENVOLVIMENTO DE SISTEMAS**

**PROFESSOR:**

**RENATO JARDIM PARDOCCI**

PROFRENATO.PARDOCCI@FIAP.COM.BR

[Renato Parducci - YouTube](#)

# APRESENTAÇÃO DO PROFESSOR

## | SHORT BIO



RENATO JARDIM PARLUCCI

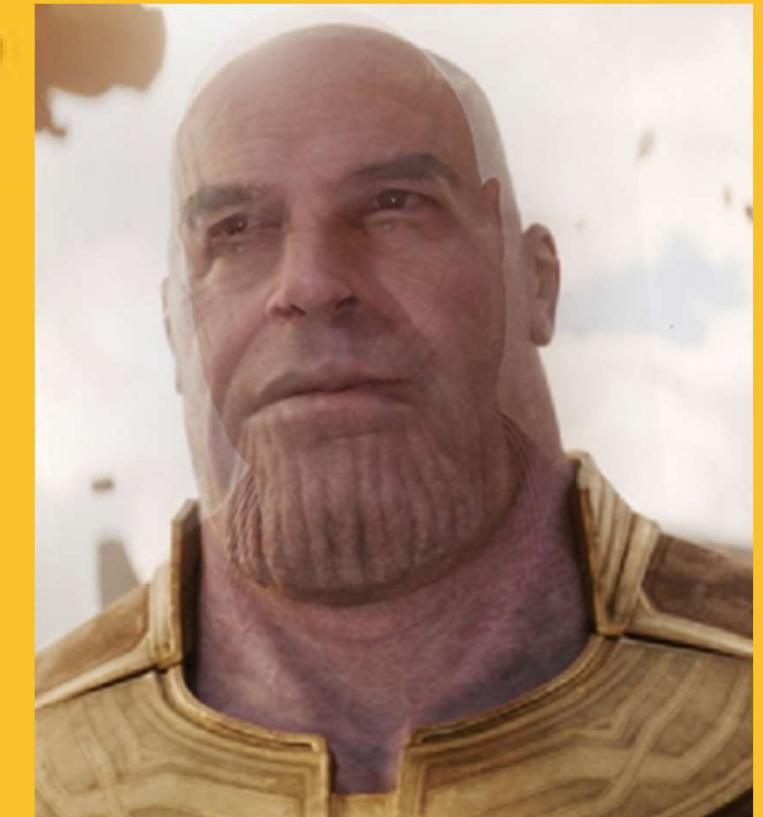
Profrenato.parducci@fiap.com.br

Apresentações

Prof. Renato Jardim Parducci



Astro nas horas extras



CREATED USING  
**PowToon**



## PREPARAÇÃO PARA MINISTRAR A DISCIPLINA

- **Membro da equipe de desenvolvimento de sistemas e teste de software da CETIL, Leão Engenharia e BG&C.**
  - **Líder de desenvolvimento de sistemas responsável pela implantação de metodologia de modelagem na Itautec, Philco.**
  - **Gestor da área de produção de software e bancos de dados na OESP-Bell South e no grupo O Estado de S. Paulo.**
  - **Gestor de infraestrutura e serviços de TI, Redes e Telecomunicações e administração de Bancos de Dados na Leão Engenharia, BG&C, OESP-Bell South, Estadão, EMS-Pharma**
  - **Consultor e gestor de implantação de programas de Qualidade de software, Fábrica de software, Escritório de projetos, Centrais de operações de TI e Suporte técnico, programas de Governança, Planejamento e estruturação estratégica de TI – Netpartners, Grupo Linx, Animatech, EMS-Pharma**
  - **Gerente de TI nas empresas Unilever, OESP Mídia e Gráfica, Estadão, Rádio Eldorado, Agência Estado, EMS Pharma, Legrand, Germed, Nova Química, ACS incorporadora.**
- Diretor de TI nas empresas Netpartners e LINX.**
- Diretor sócio nas empresas RUNAK Tecnologia e RUNMídia**

Formação



Bacharel em Ciência da Computação



Especialista em Adm. Industrial



Mestre em Eng.Computação

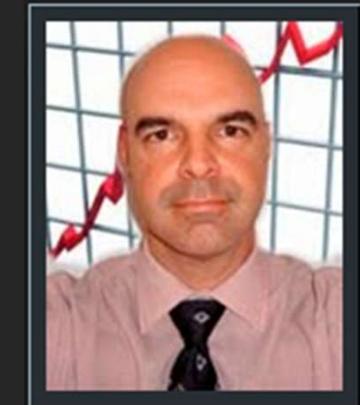


Muita experiência para trocar!

- ❖ Analista de negócios, de sistemas e desenvolvedor de software:



Executivo da área de TI:



Unilever



❖ Consultor em projetos de diversas empresas renomadas



Banco Mercedes-Benz  
DaimlerChrysler Bank



Fundação  
Carlos Chagas



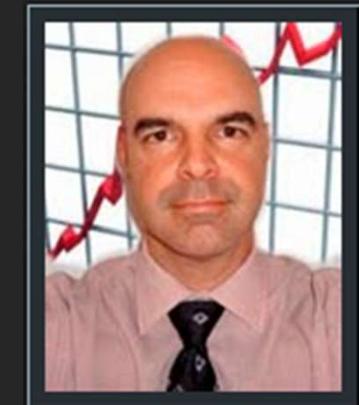
Companhia Siderúrgica Nacional



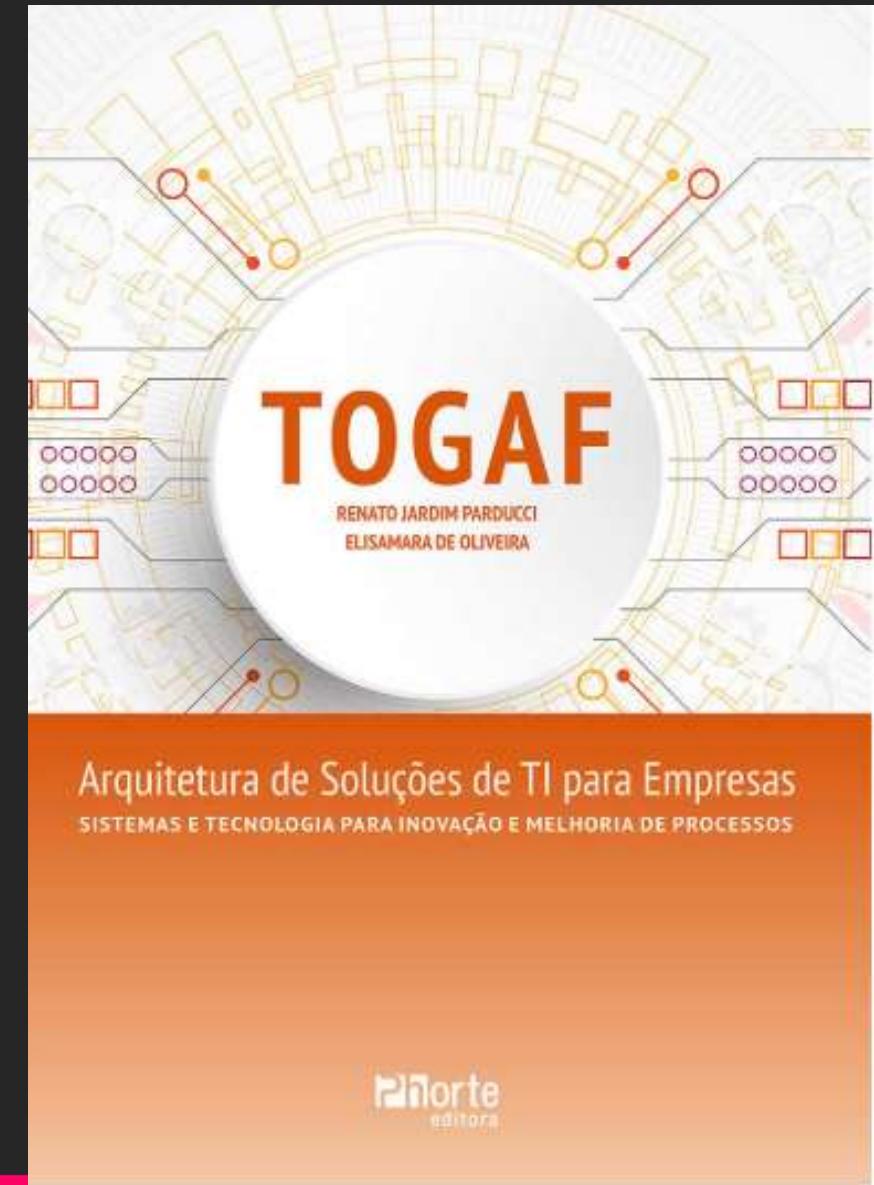
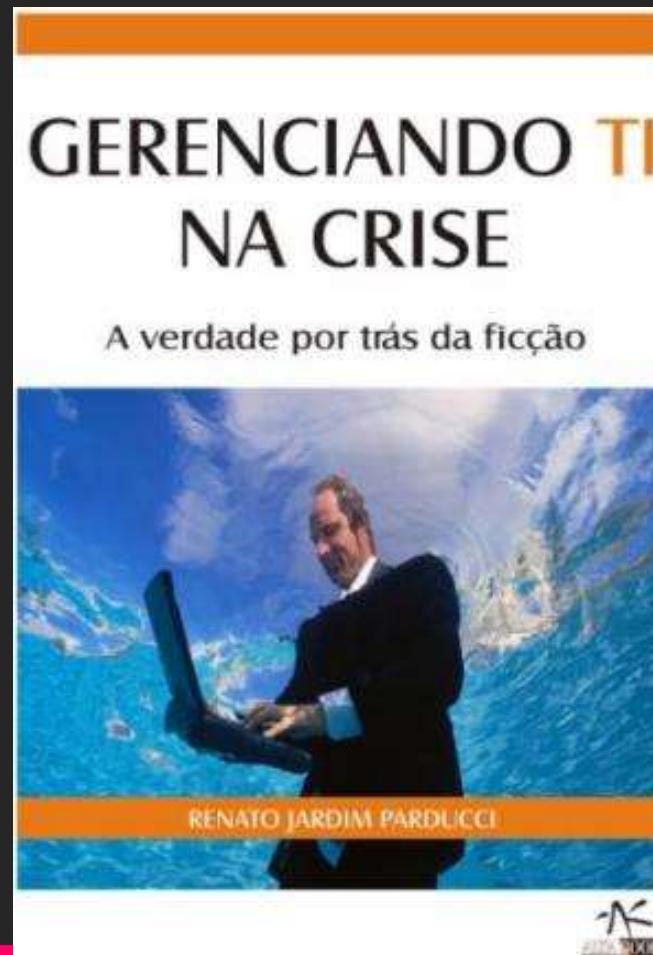
❖ Mentor de Startups



❖ Atividade acadêmica intensa



## PUBLICAÇÕES





## MÍDIA SOCIAL



- ❖ Canal do professor no Youtube
- ❖ <https://www.youtube.com/c/RenatoParducci>





# Apresentação da Disciplina

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

**Sistemas de informação e aplicativos dirigidos já são parte da vida humana e das empresas**



ERP – Sistema de gerenciamento de atividades empresariais

SIG – Sistema de informação gerencial

BI – Inteligência de negócio digital

SAD – Sistema de apoio à tomada de decisão

SE e IA – Sistema especialista que contém inteligência artificial e direciona decisões

IHM – Interfaces homem-máquina

Robótica e automação – Substituição do ser humano por máquinas em suas atividades

Data Science – Análise e tomada de decisão com base em dados heterogêneos e holísticos sobre pessoas e organizações

Realidade virtual – Interatividade entre o mundo físico e virtual

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

**Segundo o Gartner Group, estamos diante do ...**

## METAVERSO:

<https://youtu.be/clezMiOhW-0>



é um mundo virtual, criado pela união da realidade física e as atividades digitais, persistindo dados que incluem a soma de transações digitais, realidade aumentada e a comunicação integrada".

Gartner indica cinco tecnologias emergentes impactantes a longo prazo (smartplanet.pt)

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

**Segundo Max Tegmark, tudo caminha para a Vida 3.0:**



A Inteligência Artificial e as Máquinas participam da vida humana, competem pelos mesmos espaços de trabalho, influenciam decisões mais profundas, gerenciam a segurança mundial, transações financeiras, movimentações de mercadorias e demais operações do cotidiano.

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

**Com a evolução desse cenário...**

Governos, empresas e pessoas terão a possibilidade de conhecer a todas as pessoas, seus comportamentos, desejos, conhecimentos e habilidades, de forma integrada.

O anonimato e a restrição à informação demanda um elevado grau de segurança e deve estar protegida por lei.

A ética nas relações de trabalho e no tratamento da pessoa humana precisam fazer parte de programas sociais e governamentais.

**A qualidade dos dados, do software e hardware aplicado deve ser inquestionável para garantir a continuidade da vida humana e dos negócios.**

Nosso desafio é garantir que os sistemas de informação sejam aderentes às necessidades, relevantes, úteis, corretos e infalíveis.

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

## ESCOPO DA DISCIPLINA

### GESTÃO DA QUALIDADE E GOVERNANÇA NA PRODUÇÃO DE SOFTWARE

- .Qualidade de produto e governança no processo produtivo de software
- .Guias, normas e o programa da qualidade de software
- .Arquitetura de soluções tecnológicas

### GESTÃO DE PROJETOS ÁGEIS

- .Gestão integrada de projeto ágil
- .Controle de versões e mudanças em projetos

### AVALIAÇÃO DE SOFTWARE

- .Métricas de estimativa e de qualidade de desenho
- .Teste de software
- .Desenvolvimento orientado a testes e comportamento

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

## NOSSA META

<b>Objetivos</b>	<ul style="list-style-type: none"><li>Desenvolver as competências gerenciais sobre o processo de produção de software, com objetivos de garantir a qualidade do produto final e o desempenho adequado do processo de negócio associado à aplicação do sistema informatizado.</li><li>Conhecer e aplicar controles sobre os processos de produção de software para administrar adequadamente a eficácia, eficiência e efetividade e gerar um produto de software que cumpra com os objetivos da qualidade, atendendo as necessidades de patrocinadores, clientes e equipes que cuidarão da sustentação do produto.</li><li>Adquirir conhecimento fundamental para participar de processos de auditoria da qualidade para certificação ou dentro de um programa de melhoria contínua.</li></ul>
<b>Competências</b>	<ul style="list-style-type: none"><li>Desenvolver o plano de qualidade dentro de um planejamento de projetos integrado, de forma a garantir a governança.</li><li>Utilizar métricas quantitativas para estimar projetos de software.</li><li>Gerenciar o processo produtivo de software com métricas e controle estatístico de processos.</li><li>Aplicar métricas de avaliação estrutural do modelo de software.</li><li>Planejar, projetar e aplicar testes de produto, cumprindo níveis, tipos e técnicas específicas que atendem os princípios de BDD (<u>Behavior Driven Development</u>) e TDD (<u>Test Driven Development</u>).</li></ul>

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

## NOSSA META

### Habilidades

- Criar controles e diagnósticos de TI alinhados com as expectativas de negócio;
- Aumentar o nível de maturidade e capacidade em desenvolvimento de software com qualidade;
- Participar de programas de certificação da qualidade em software;
- Criar controles para gerenciamento de requisitos;
- Entender o processo de gerenciamento de requisitos;
- Entender e identificar as etapas do processo de medição funcional e estrutural do software;
- Calcular tamanho de software de acordo com seus requisitos funcionais, possibilitando estimativas de resultado de esforço, custo e prazo;
- Diferenciar níveis de teste, técnicas e tipos de teste;
- Utilizar técnicas para planejar, aplicar e registrar resultados de testes;
- Criar e executar casos, roteiros e planos de teste;
- Usar testes para guiar a produção do software;
- Fazer uso de ferramentas de planejamento, controle e automação teste baseadas em software e ferramentas CASE (auxílio computacional a engenharia de software);
- Aplicar de forma integrada as práticas de gerenciamento de projetos com as práticas da qualidade.

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

## DETALHES DO CONTEÚDO

### 1º Semestre:

- Governança de TI com COBIT
- Gestão da qualidade total em TI com ISO, CMMI, MPS.br
- Alinhamento estratégico de TI , arquitetando soluções com TOGAF
- Processo Ágil de Software com SCRUM
- Gestão da produção integrada e mudanças de software
- Estimativa paramétrica e o gerenciamento quantitativo de projetos

### 2º Semestre

- Processo de teste de software
- Ferramentas e práticas de planejamento de testes
- TDD – Desenvolvimento de software orientado a testes
- Ferramentas e práticas de testes manuais de software
- Automação de testes
- Data Quality

Aulas práticas no modelo “hands on”, orientadas por projetos e estudos de caso

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

## MATERIAL DE AULA DE APOIO

### SLIDES DE AULA

.Material didático com teoria e exemplificação prática

### ESTUDOS DE CASO

.Material contendo desafios e soluções realizados em sala de aula, com base em estudo de caso prático

### EXERCÍCIOS

.Lista de exercícios resolvidos (com resposta no próprio material) e propostos (desafios para o estudante que não têm resposta no material)

### JOGOS DE FIXAÇÃO

### CANAL DO PROFESSOR

.Vídeos sobre ferramentas e métodos estudados, com link nos slides de aula

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

## BIBLIOGRAFIA

Bibliografia	
<b>Básica</b>	<ol style="list-style-type: none"><li>1- <b>PRESSMAN</b>, R. S.; <b>Maxim</b>, B. R. Engenharia de Software – Uma abordagem Profissional. 7ª ed. Porto Alegre: Editora Bookman, 2011.</li><li>2- <b>MANSUR</b>, R. Governança da nova TI. A Revolução. Rio de Janeiro: Editora Ciência Moderna, 2013.</li><li>3- <b>SELEME</b>, Robson, <b>STADLER</b>, Humberto. <b>Controle da Qualidade - As ferramentas essenciais</b>. 1ª ed. <u>Intersaberes</u>, 2012. *</li></ol>
<b>Complementar</b>	<ol style="list-style-type: none"><li>1- <b>PFLEEGER</b>, Shari Lawrence. <b>Engenharia de software: teoria e prática</b>. 2ª ed. São Paulo: Pearson <u>Education</u> do Brasil, 2004. *</li><li>2- <b>FOGGETTI</b>, C. (organizador) <b>Gestão Ágil de Projetos</b>. São Paulo: Pearson <u>Education</u> do Brasil, 2014. *</li><li>3- <b>LÉLIS</b>, Eliacy Cavalcanti. (organizador) <b>Gestão da Qualidade</b>. 1ª ed. São Paulo: Pearson <u>Education</u> do Brasil 2012. *</li><li>4- <b>ISACA. COBIT 5, USA, 2014</b> - Disponível em: <a href="http://www.isaca.org/cobit/pages/default.aspx">http://www.isaca.org/cobit/pages/default.aspx</a>. *</li><li>5- <b>SEI, Carnegie Mellon University. CMMI V3. SEI - Software Engineering Institute., USA, 2007</b>. Disponível em: <a href="https://www.sei.cmu.edu/cmmi/">https://www.sei.cmu.edu/cmmi/</a> *</li></ol>

(\*) Acervo online

Outras bibliografias poderão ser indicadas nos materiais de apoio

# GOVERNANÇA E MELHORES PRÁTICAS EM TI

Espaço para você manifestar as suas expectativas e se apresentar!



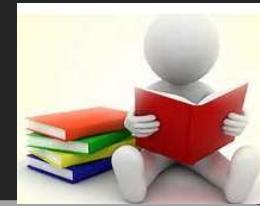


VAMOS  
COMEÇAR!

## BOAS PRÁTICAS NA PRÁTICA

Vamos trabalhar um **estudo de caso** para estudar e aplicar boas práticas no desenvolvimento de sistemas de informação.

## ESTUDO DE CASO SIMULADO



Dilan, o dono de uma software house tem visto muita reclamação de clientes que contratam seus projetos de desenvolvimento de sistemas. Ele acredita que uma ferramenta mais eficiente para testar software possa eliminar grande parte dos seus problemas com os clientes.

Como a empresa faz programação em linguagem JAVA, Consuelo, uma consultora contratada para ensinar boas práticas em desenvolvimento de sistemas, indicou o uso da ferramenta JUNIT que é integrada ao Eclipse.

Consuelo preparou um treinamento para aprendizado passo a passo do uso de JUNIT para criar scripts de teste que possam ser executados quantas vezes forem necessários, criando um autômato de teste.

Vamos seguir as instruções do programa de treinamento de Consuelo que vem a seguir, a qual inicial com uma pequena introdução sobre testes para conscientizar os desenvolvedores.

## GERENCIAMENTO DO TESTE DE SOFTWARE

Teste é uma atividade exaustiva e muitas vezes pouco interessante aos olhos do programador mas,... Quando testar se torna também uma atividade de programação, tudo fica mais divertido ( os programadores ficam estimulados a testar )!

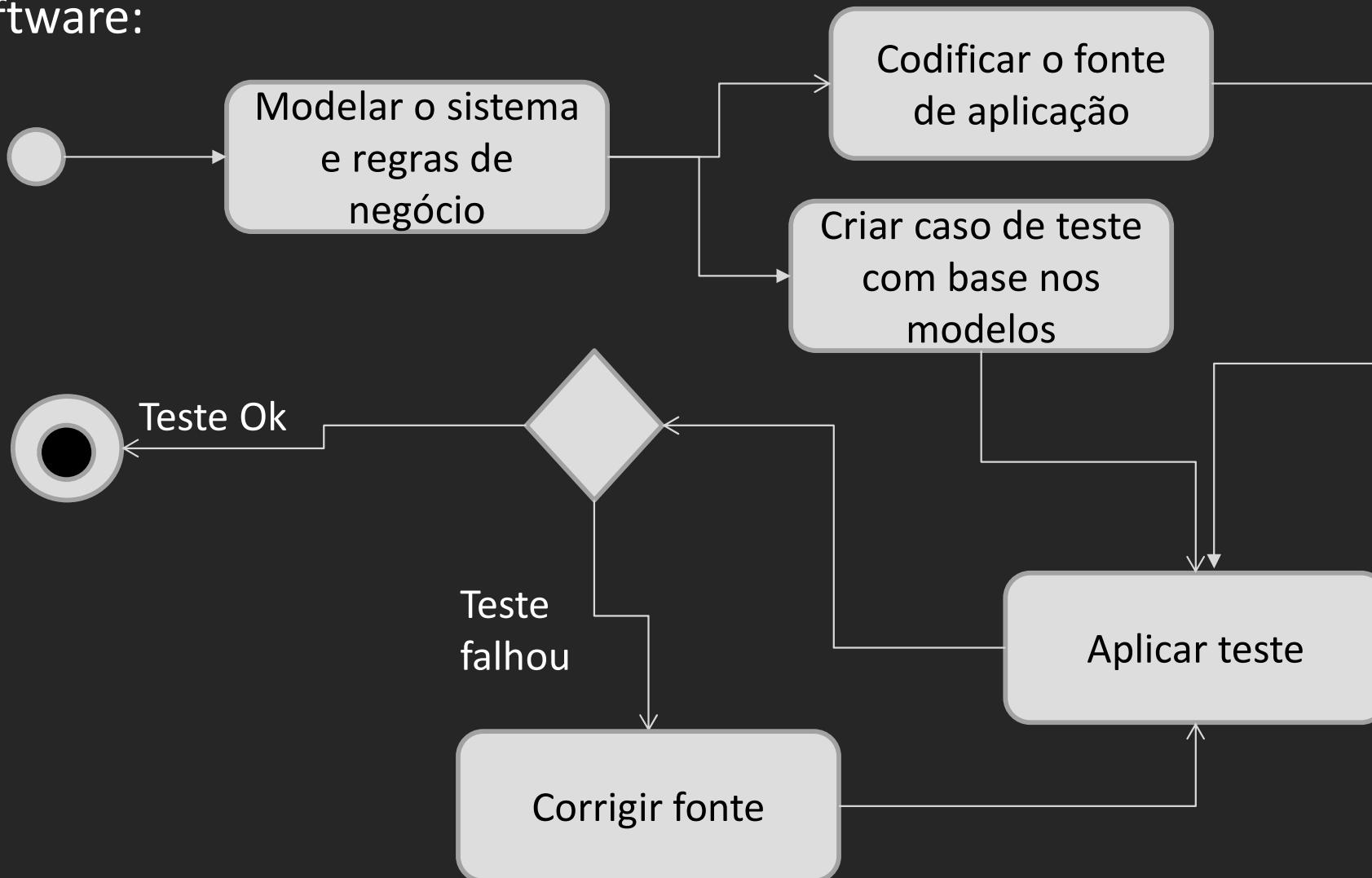
## GERENCIAMENTO DO TESTE DE SOFTWARE

Utilizar técnicas apropriadas e ferramentas adequadas para testar programas de aplicação permite:

- Disciplinar o formato dos testes;
- Documentar os casos de teste;
- Aplicar um mesmo teste múltiplas vezes, se necessário (repetir);
- Aproveitar um teste criado por um programador por outras pessoas do time de desenvolvimento (reusar);
- Agilizar a execução dos casos de testes e avaliação dos resultados;
- Avaliar se os testes criados cobrem as situações previstas na lógica de um programa de aplicação (análise de cobertura dos testes).

## GERENCIAMENTO DO TESTE DE SOFTWARE

Vamos trabalhar neste momento com o **processo convencional de teste de software**:



## GERENCIAMENTO DO TESTE DE SOFTWARE

Para criar os casos de testes, vamos utilizar a ferramenta aplicada a programas JAVA:

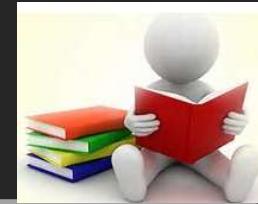


## GERENCIAMENTO DO TESTE DE SOFTWARE

Vamos usar JUNIT dentro do Eclipse, aprendendo na prática a criar os testes automatizados:



## ESTUDO DE CASO SIMULADO



Você contou para Consuelo que não poderá participar do treinamento, por conta de estar ocupado com o desenvolvimento e entrega “pra ontem” de um programa de aplicação de calculadora digital.

Consuelo, então, propôs que seja usado o seu programa como exemplo de como usar JUNIT em testes.

Excelente para você!  
Vai aprender e ganhar tempo!

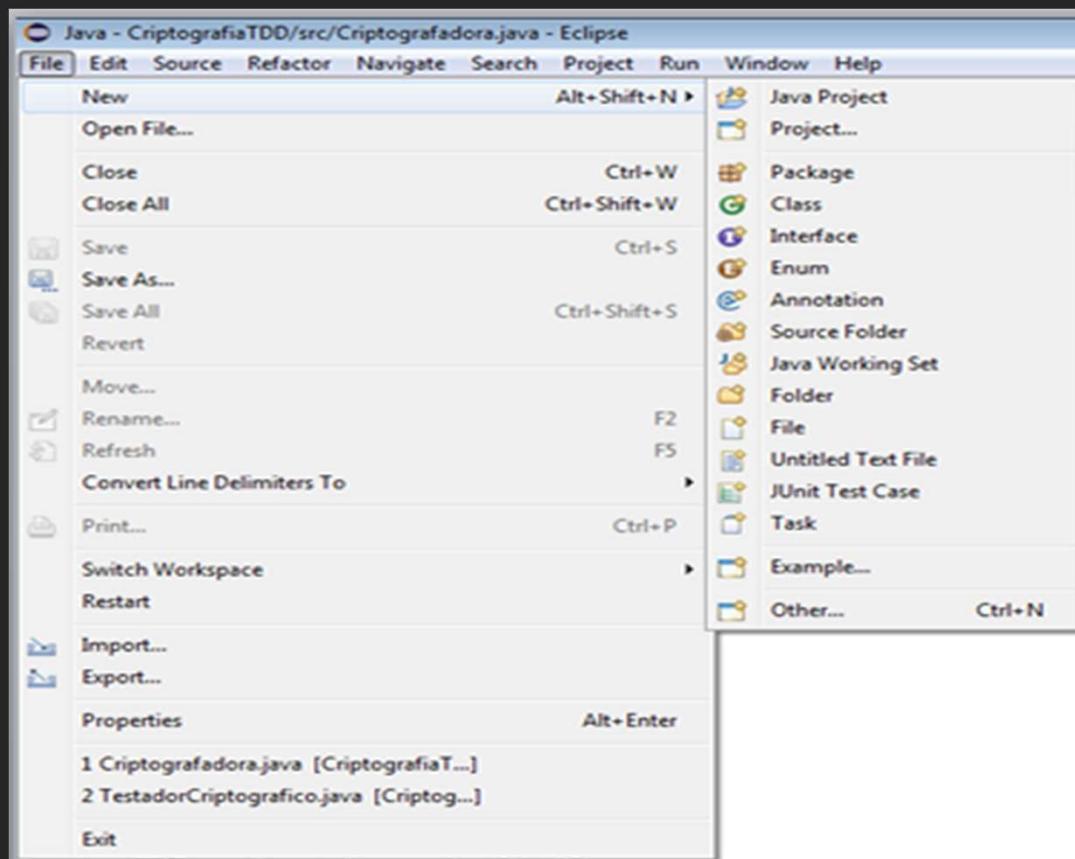
## GERENCIAMENTO DO TESTE DE SOFTWARE

Vamos iniciar pela criação de uma Classe JAVA para trabalharmos os testes.

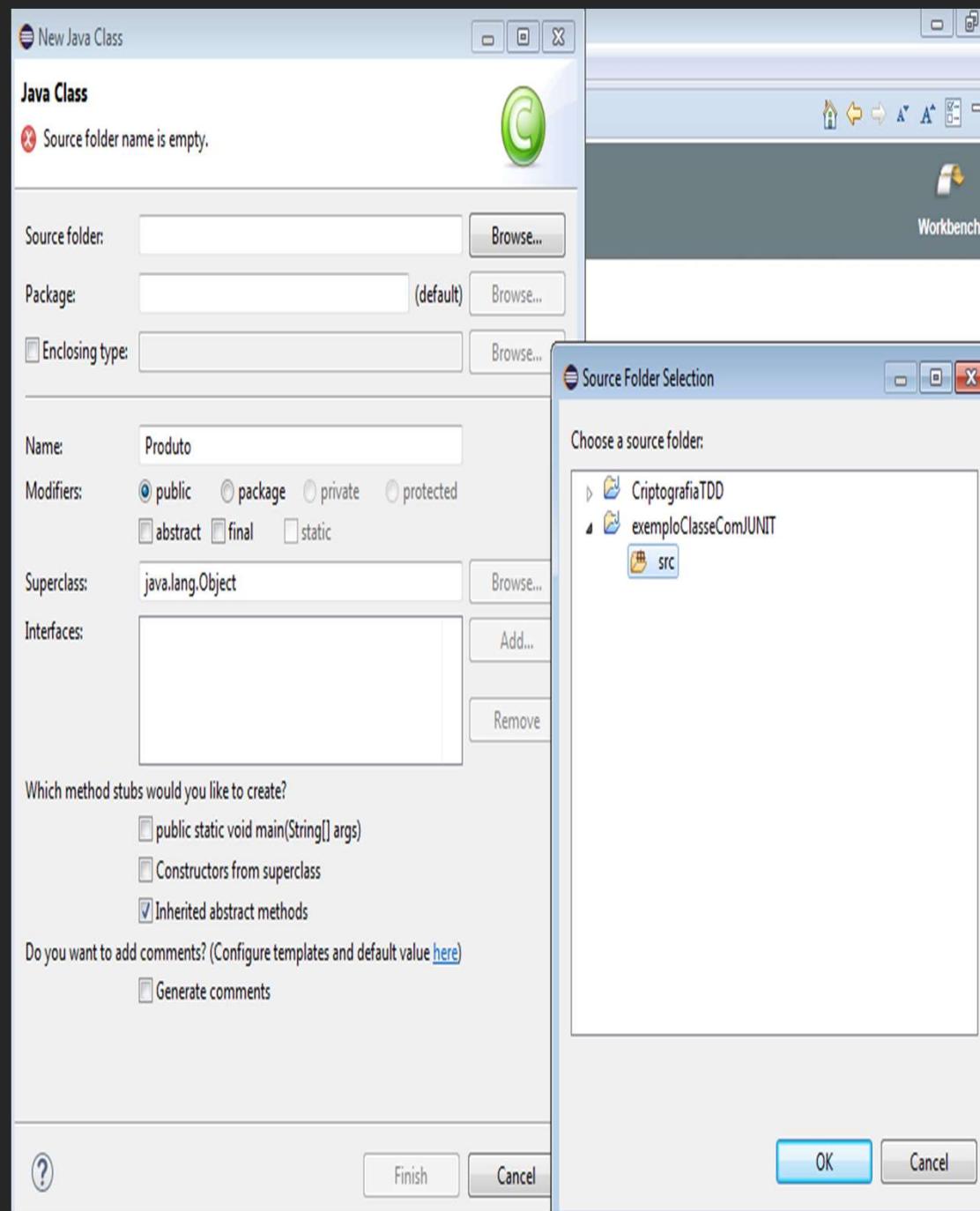
Nos exemplos a seguir, é considerado que a Classe foi escrita exatamente conforme os algoritmos de especificação e modelo UML que foram definidos para seus atributos e métodos.

Queremos confirmar que ela funciona adequadamente, através dos testes.

Crie um projeto no Eclipse e depois, ...  
a Classe JAVA Calculadora, descrita ao  
lado (*new JAVA Class EJB*)



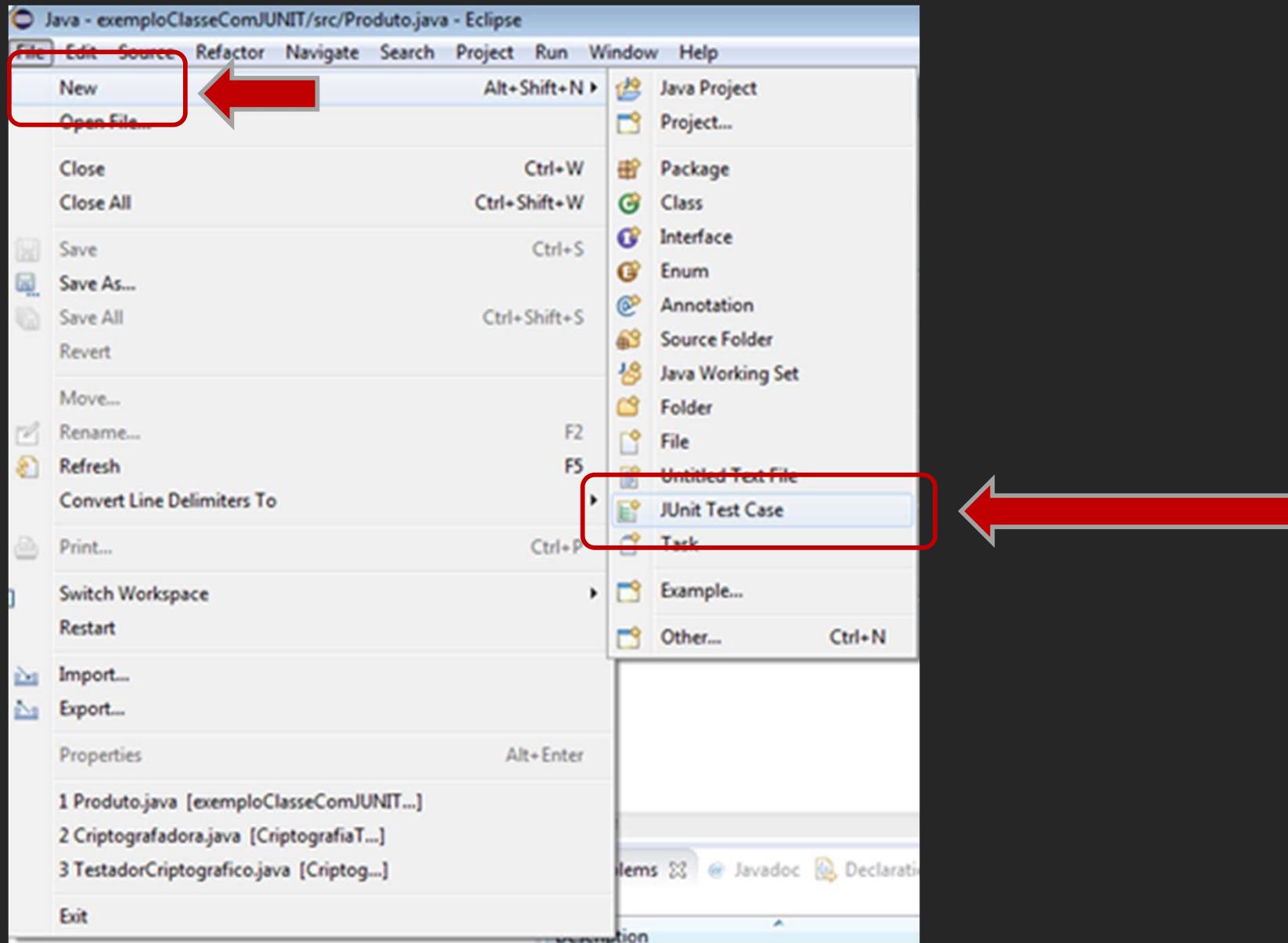
```
public class Calculadora{  
  
    // atributo  
    private int resultado = 0;  
  
    // método somar  
    public int somar( int n1, int n2 ){  
  
        resultado = n1 + n2;  
        return resultado;  
    }  
  
    // método subtrair  
    public int subtrair( int n1, int n2 ){  
  
        resultado = n1 - n2;  
        return resultado;  
    }  
  
    // método multiplicar  
    public int multiplicar( int n1, int n2 ){  
  
        resultado = n1 * n2;  
        return resultado;  
    }  
  
    // método dividir  
    public int dividir( int n1, int n2 ){  
  
        resultado = n1 / n2;  
        return resultado;  
    }  
}
```



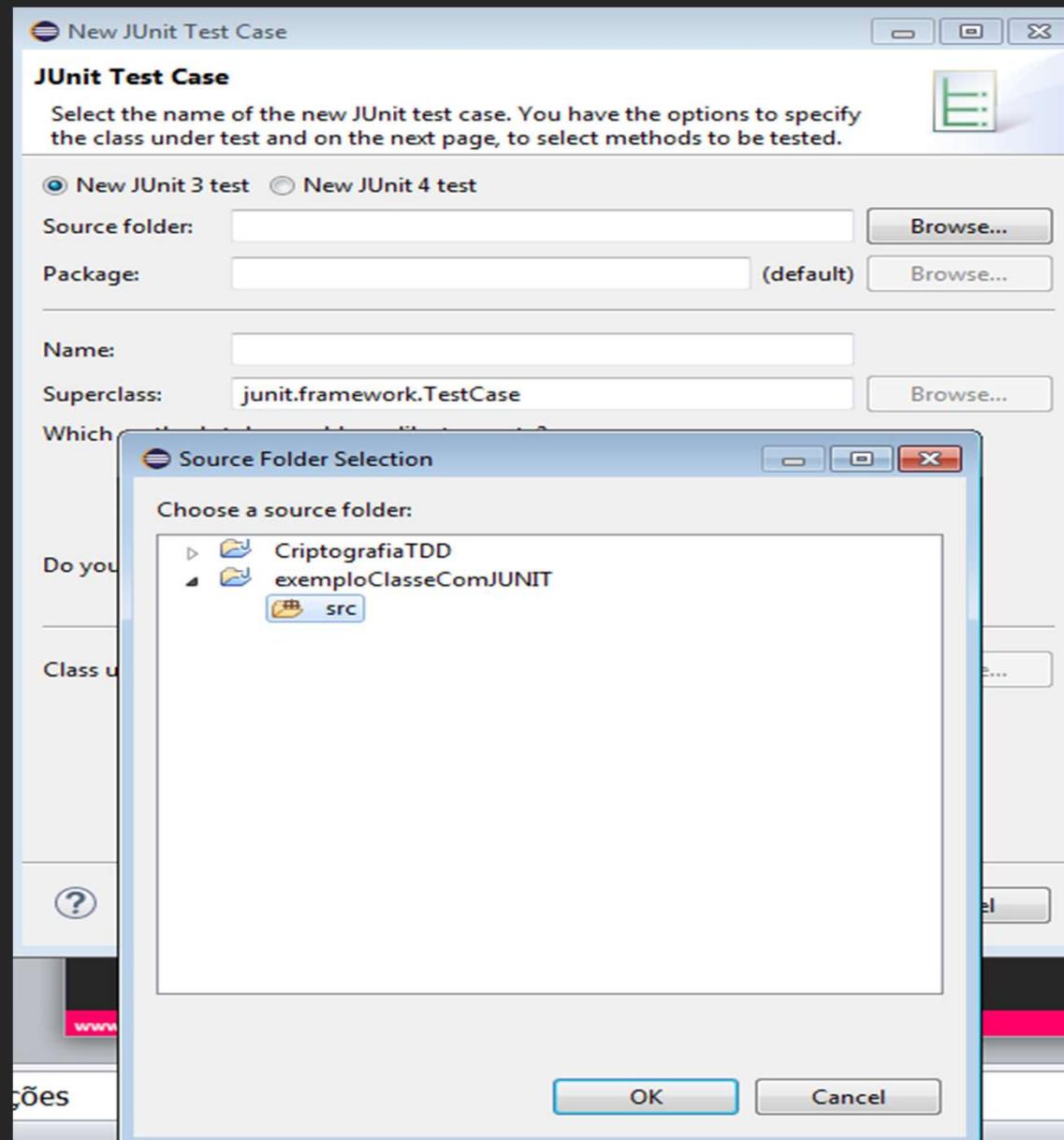
```
public class Calculadora{  
  
    // atributo  
    private int resultado = 0;  
  
    // método somar  
    public int somar( int n1, int n2 ){  
  
        resultado = n1 + n2;  
        return resultado;  
    }  
  
    // método subtrair  
    public int subtrair( int n1, int n2 ){  
  
        resultado = n1 - n2;  
        return resultado;  
    }  
  
    // método multiplicar  
    public int multiplicar( int n1, int n2 ){  
  
        resultado = n1 * n2;  
        return resultado;  
    }  
  
    // método dividir  
    public int dividir( int n1, int n2 ){  
  
        resultado = n1 / n2;  
        return resultado;  
    }  
}
```

Agora, vamos aos testes...

Crie uma JUNIT Test Case (new JUNIT)



Dê o nome de TesteCalculadora para a Classe de teste que será criada.



## GERENCIAMENTO DO TESTE DE SOFTWARE

O processo que vamos seguir para realizar a criação dos testes será:

- 1º) Criar uma Classe de Teste para Classe de Implementação;
- 2º) Criar um Método de Teste diferente dentro da Classe de Teste para cada simulação de comportamento que for necessária (um Método para cada Caso de Teste).
- 3º) Criar e aplicar um Método de Teste por vez, isolando problemas (keep it simple – faça as coisas de forma simples).

```
import static org.junit.Assert.assertEquals;  
  
import junit.framework.TestCase;  
  
public class TesteCalculadoraTest extends TestCase {  
  
    /**  
     * Teste de somar na Calculadora.  
     */  
    @Test  
    public void testeSomar() {  
        int nro1 = 5;  
        int nro2 = 5;  
        Calculadora calc= new Calculadora();  
        int resultadoEsperado = 10;  
        int resultadoReal= calc.somar(nro1, nro2);  
        assertEquals(resultadoEsperado, resultadoReal);  
    }  
}
```

Implementação do Método de teste SOMA, dentro da Classe de Teste de um Objeto Calculadora

```
import static org.junit.Assert.assertEquals;  
import junit.framework.TestCase;  
  
public class TesteCalculadoraTest extends TestCase {  
  
    ...  
  
    /**  
     * Teste de subtrair na Calculadora.  
     */  
    @Test  
    public void testeSubtrair() {  
        int nro1 = 5;  
        int nro2 = 3;  
        Calculadora calc = new Calculadora();  
        int resultadoEsperado= 2;  
        int resultadoReal= calc.subtrair(nro1, nro2);  
        assertEquals(resultadoEsperado, resultadoReal);  
    }  
}
```

Acrescente esse Método de teste da SUBTRAÇÃO logo em seguida do Método de teste de SOMA que você fez anteriormente

```
import static org.junit.Assert.assertEquals;  
import junit.framework.TestCase;  
  
public class TesteCalculadoraTest extends TestCase {  
  
    ...  
  
    /**  
     * Teste de multiplicar na Calculadora.  
     */  
    @Test  
    public void testeMultiplicar() {  
        int nro1 = 3;  
        int nro2 = 3;  
        Calculadora calc = new Calculadora();  
        int resultadoEsperado = 9;  
        int resultadoReal = calc.multiplicar(nro1, nro2);  
        assertEquals(resultadoEsperado, resultadoReal);  
    }  
}
```

Acrescente esse Método de teste da MULTIPLICAÇÃO logo em seguida do Método de teste de SUBTRAÇÃO que você fez anteriormente

```
import static org.junit.Assert.assertEquals;  
import junit.framework.TestCase;  
  
public class TesteCalculadoraTest extends TestCase {  
  
    ...  
  
    /**  
     * Teste de dividir na Calculadora.  
     */  
    @Test  
    public void testeDividir() {  
        int nro1 = 6;  
        int nro2 = 2;  
        Calculadora calc = new Calculadora();  
        int resultadoEsperado= 3;  
        int resultadoReal = calc.dividir(nro1, nro2);  
        assertEquals(resultadoEsperado, resultadoReal);  
    }  
}
```

Acrescente esse Método de teste da DIVISÃO logo em seguida do Método de teste de MULTIPLICAÇÃO que você fez anteriormente

```
public class Produto{
```



```
private double peso;  
private double altura;
```

```
public double getPeso() {  
    return peso;  
}
```

```
public void setPeso(double peso) {  
    this.peso = peso;  
}
```

```
public double getAltura() {  
    return altura;  
}
```

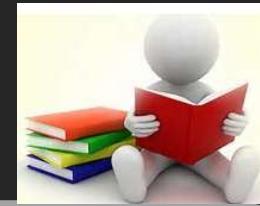
```
public void setAltura(double altura) {  
    this.altura = altura;  
}
```

```
}
```

Agora, crie a Classe descrita ao lado e a JUNIT para testar todos os métodos da Classe.

*Faça os testes para criar um objeto e instanciá-lo e depois testar a recuperação de dados.*

## ESTUDO DE CASO SIMULADO



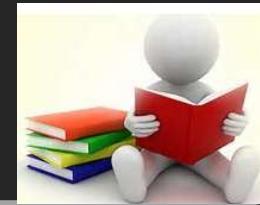
Após o treinamento, um dos participantes falou que muitas vezes, uma aplicação que está desenvolvendo envolve validar várias classes, uma a uma.

Ele quer saber se é possível usar a JUNIT para fazer vários testes simultâneos, de forma organizada.

Uma preocupação adicional é poder ajustar o ambiente de teste (situação de tabelas, conexão de banco, iniciação de variáveis, antes e depois de cada caso de teste aplicado, de forma a evitar que “lixo” (instâncias) de um teste feito, contaminem um teste seguinte.

Consuelo apontou que esses recursos existem e vai usar um programa que está em desenvolvimento pela equipe para explicar como funcionam esses recursos.

## ESTUDO DE CASO SIMULADO

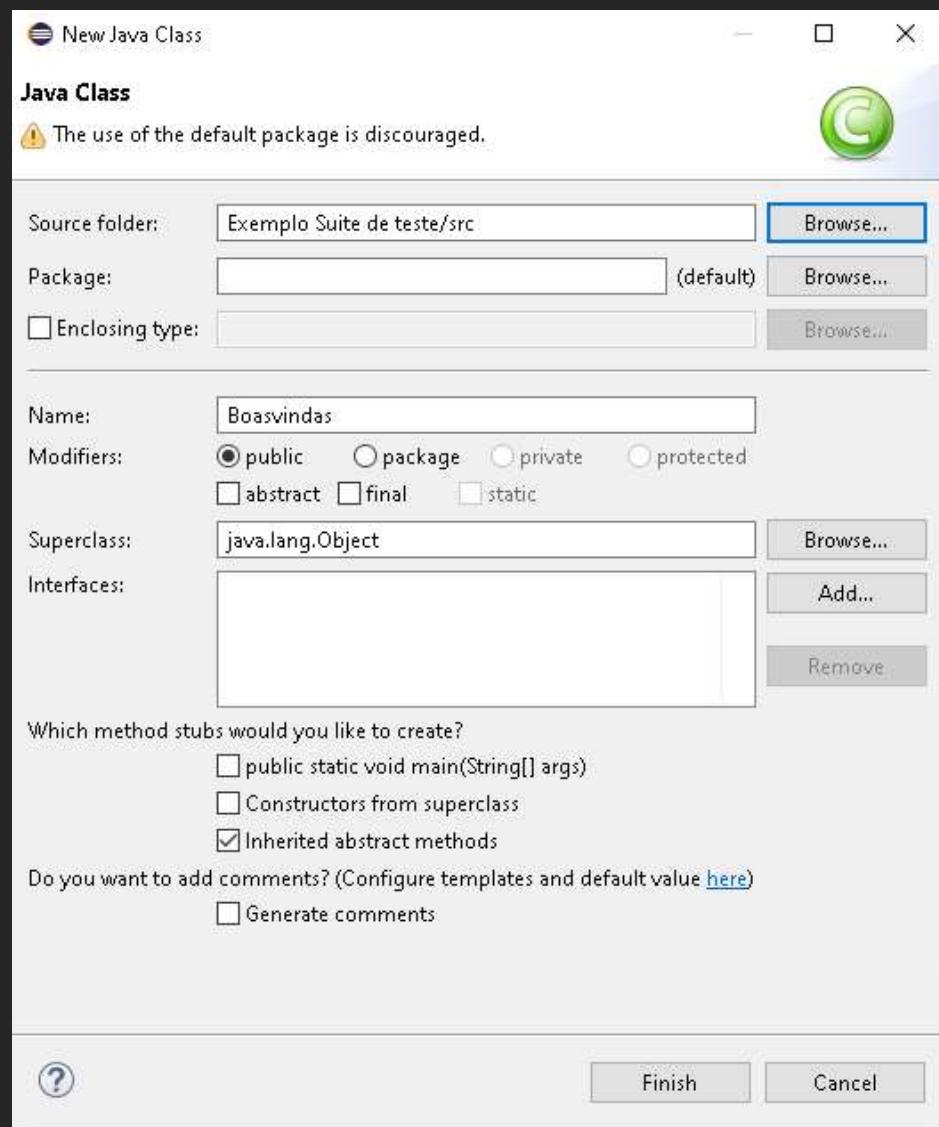


1º) Crie os testes para a Classe a seguir, a qual vai ser validada junto com a Calculadora que você acabou de desenvolver!

Essa Classe, exibe uma mensagem de boas vindas ao usuário da Calculadora digital.

Crie essa Classe dentro do mesmo projeto da Calculadora!

Crie um projeto no Eclipse e depois, ...  
a Classe JAVA para exibir mensagem de boas vindas customizada ao usuário  
de um sistema



```
public class Boasvindas{  
  
    private String mensagem;  
  
    //Construtor de Objeto na Classe  
    public Boasvindas(String mens){  
        this.mensagem = mens;  
    }  
  
    // Exibição da mensagem  
    public String exibirMensagem(){  
        System.out.println(this.mensagem);  
        return this.mensagem;  
    }  
  
    // Exibição da parte fixa da mensagem  
    public String completarMensagem(){  
        String compmens;  
        compmens = "Olá! Seja bem vindo à sua calculadora pessoal"  
        System.out.println(compmens);  
        return compmens;  
    }  
}
```

Agora, vamos criar os testes para essa classe!

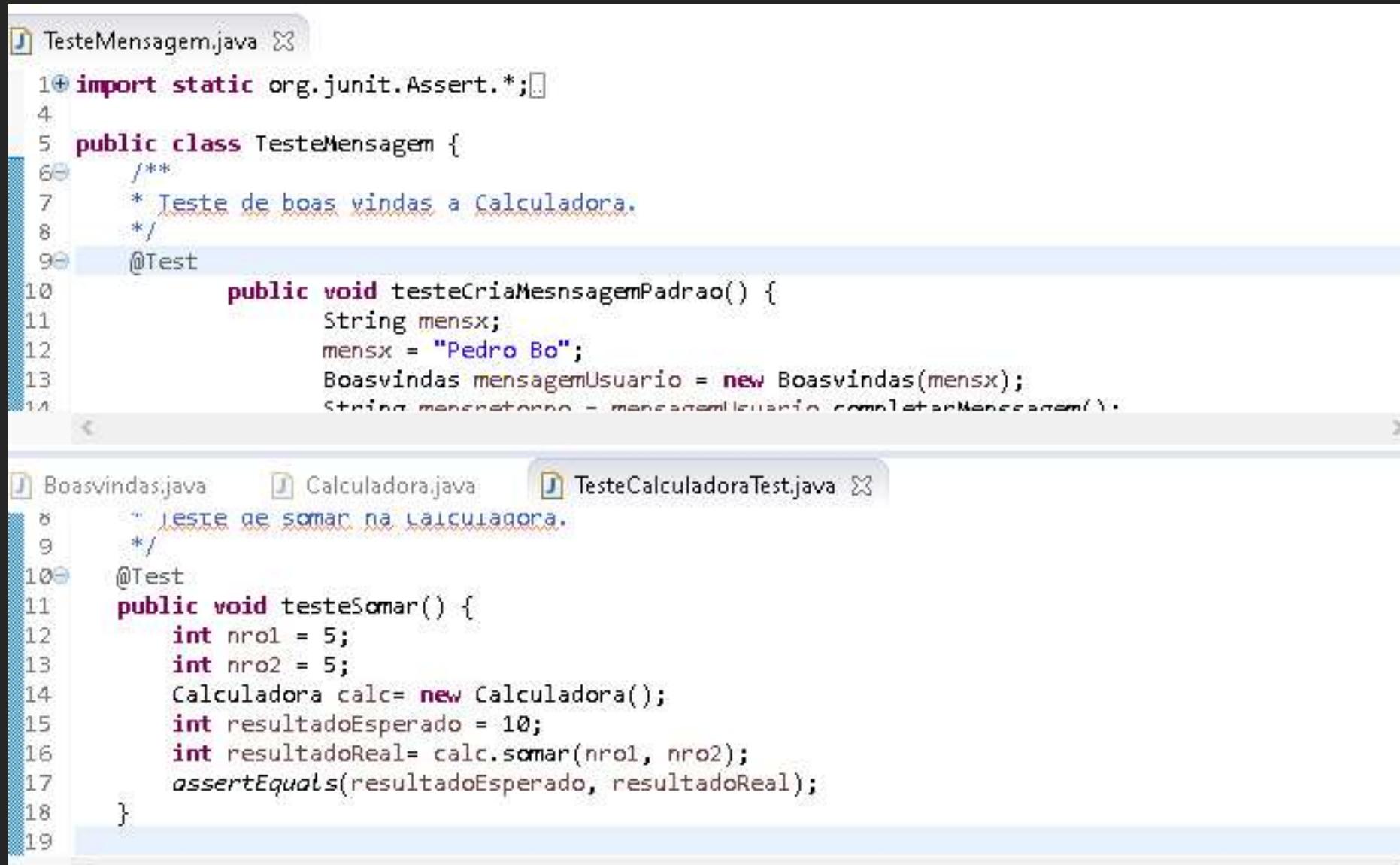
```
import static org.junit.Assert.*;
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class TesteMensagem {

    @Test
    public void testeCriaMesnsagemPadrao() {
        String mensx;
        mensx = "Pedro Bo";
        Boasvindas mensagemUsuario = new Boasvindas(mensx);
        String mensretorno = mensagemUsuario.completarMenssagem();
        assertEquals("Ola! Seja bem vindo a sua calculadora pessoal", mensretorno);
    }

    @Test
    public void testeExibeMesnsagem() {
        String mensx;
        mensx = "Pedro Bo";
        Boasvindas mensagemUsuario = new Boasvindas(mensx);
        String mensRetorno;
        mensRetorno = mensagemUsuario.exibirMenssagem();
        assertEquals(mensx, mensRetorno);
    }
}
```

Você agora tem duas Classe e duas Classes de testes no mesmo projeto!  
*Para rodá-las todas juntas, precisará criar uma Suite de Teste!*

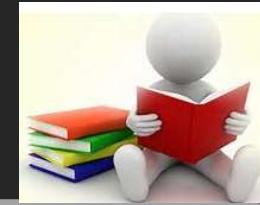


The screenshot shows an IDE interface with two code editors. The top editor contains `TesteMensagem.java` and the bottom editor contains `TesteCalculadoraTest.java`. Both files are in Java syntax highlighting mode.

```
TesteMensagem.java
import static org.junit.Assert.*;
public class TesteMensagem {
    /**
     * Teste de boas vindas a Calculadora.
     */
    @Test
    public void testeCriaMesnsagemPadrao() {
        String mensx;
        mensx = "Pedro Bo";
        Boasvindas mensagemUsuario = new Boasvindas(mensx);
        String mensagetoque = mensagemUsuario.completarMensagem();
    }
}

Boasvindas.java
    /**
     * Teste de somar na Calculadora.
     */
    @Test
    public void testeSomar() {
        int nro1 = 5;
        int nro2 = 5;
        Calculadora calc= new Calculadora();
        int resultadoEsperado = 10;
        int resultadoReal= calc.somar(nro1, nro2);
        assertEquals(resultadoEsperado, resultadoReal);
    }
}
```

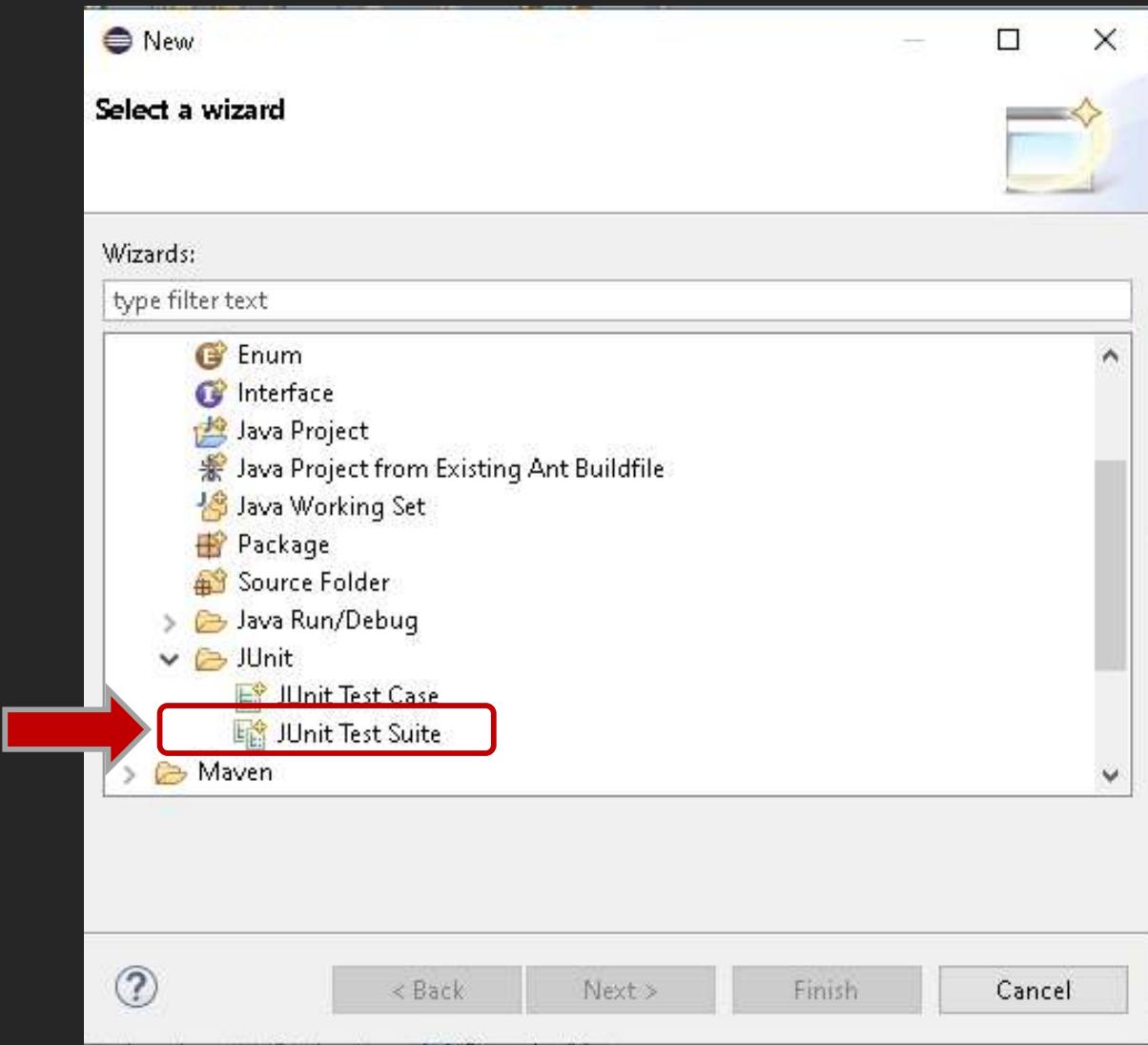
ESTUDO DE CASO SIMULADO



2º) Agora, vamos mixar esses testes com os que você criou anteriormente!

Siga os passos...

Crie a suíte de teste, selecionando o Folder do seu projeto!



Vamos renomear a suíte para TesteCompletoCalculadora e incluir as chamadas de todas as Classes JUNIT!

```
import org.junit.runner.RunWith;
```

```
import org.junit.runners.Suite;
```

```
@RunWith(Suite.class)
```

```
@Suite.SuiteClasses({
```

```
    TesteMensagem.class,
```

```
    TesteCalculadoraTest.class
```

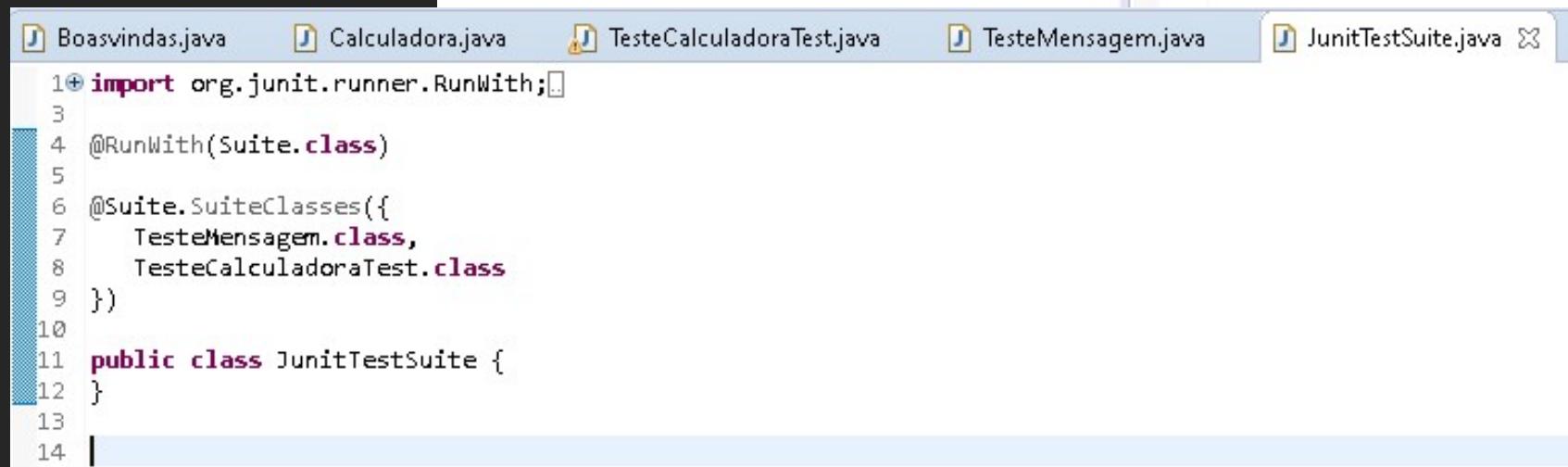
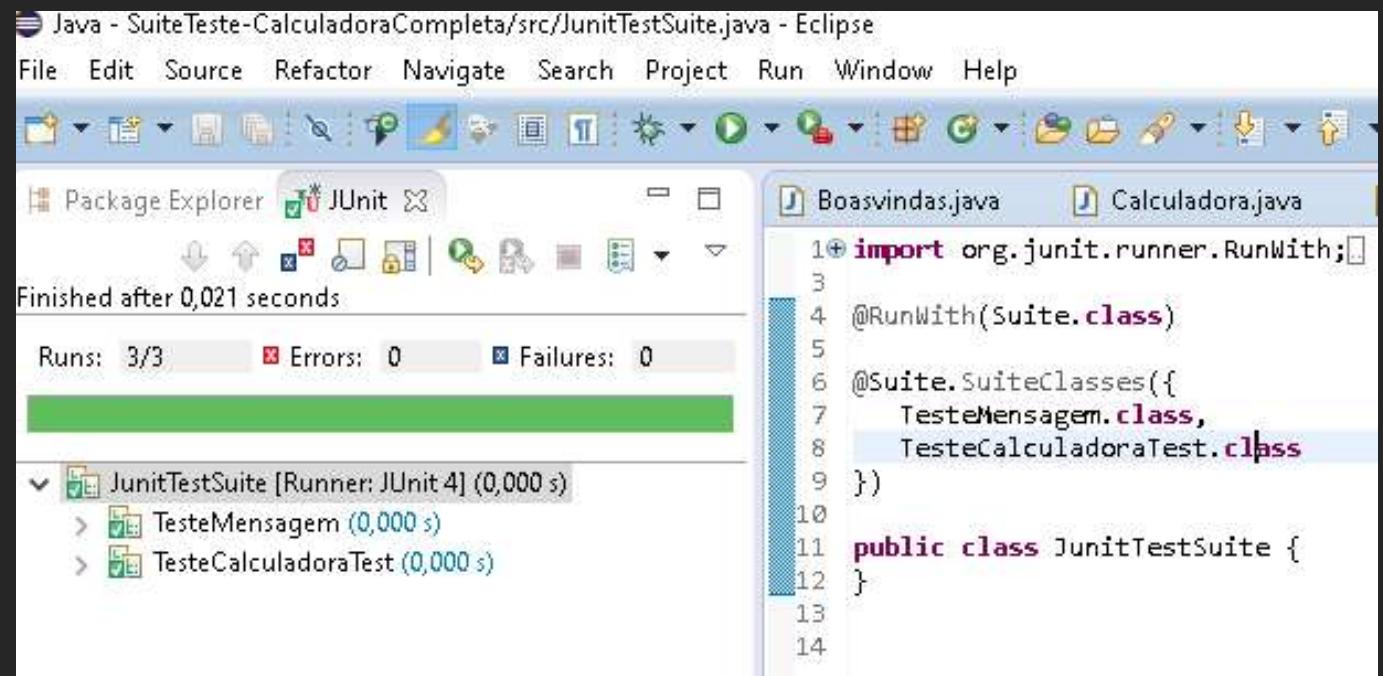
```
)
```

```
public class JunitTestSuite {
```

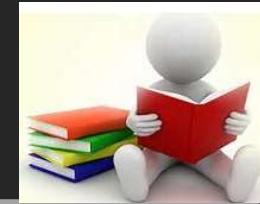
```
}
```

*Depois, é só selecionar a Suite na hora de executar os testes e ela mostrará o resultado de cada Classe/Método!*

```
import org.junit.runner.RunWith;  
import org.junit.runners.Suite;  
  
@RunWith(Suite.class)  
  
@Suite.SuiteClasses({  
    TesteMensagem.class,  
    TesteCalculadoraTest.class  
})  
  
public class JunitTestSuite {  
}
```



## ESTUDO DE CASO SIMULADO



Os desenvolvedores querem poder disparar os testes via linha de comandos (prompt)!

É possível? Perguntaram à Consuelo.

Ela respondeu que é necessário criar um job executor de testes (test runner).

Veja como fazer...

Vamos criar uma nova Classe, chamada Executoradetestes, no mesmo projeto onde está a Suite, as Classes do sistema e as Junit Classes!

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class Executoradetestes {
    public static void main(String[] args) {
        Result resultado = JUnitCore.runClasses(JunitTestSuite.class);

        for (Failure failure : resultado.getFailures()) {
            System.out.println(failure.toString());
        }

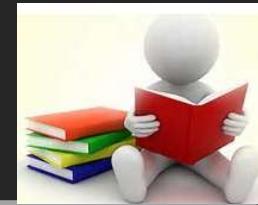
        System.out.println(resultado.wasSuccessful());
    }
}
```

*Você pode executar como uma aplicação JAVA no próprio Eclipse ou executar via linha de comando, após compilação de todas as Classes e Junits no JAVA (comando JAVAR <nome da classe>).*

```
C:\JUNIT_WORKSPACE>javac Calculadora.java Boasvindas.java  
TesteCalculadoraTest.java TesteMensagem.java Executoradetestes.java
```

```
C:\JUNIT_WORKSPACE>java Executoradetestes
```

## ESTUDO DE CASO SIMULADO



Você está em um projeto de um software e precisa criar uma função que colha 3 digitações de números inteiros, compare os três e diga qual o maior e qual o menor número digitado.

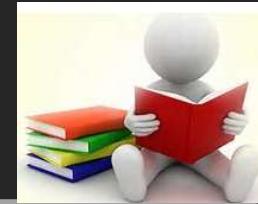
- 1º) Crie a Classe JAVA e o Método de Captura de Digitação de um número, mais o Método exibir o Maior e outro para exibir o Menor número entre três digitados.
- 2º) Crie a Classe de Testes e seus Métodos, com JUNIT.
- 3º) Crie depois a Suíte de teste.  
Crie a classe executora dos testes.
- 4º) Depois que tudo estiver funcionando, transfira a Classe e as JUNITs para o projeto que foi aplicado no treinamento (Calculadora Digital) e ajuste a Test Suite Class e a Test Runner para acomodar essa nova classe e seus testes.

*APROVEITE PARA AUTOMATIZAR TESTES UNITÁRIOS COM JUNIT EM TODOS OS SEUS PROJETOS JAVA, DAQUI POR DIANTE!*



## GESTÃO DE CONTEÚDO E VERSIONAMENTO

## ESTUDO DE CASO SIMULADO

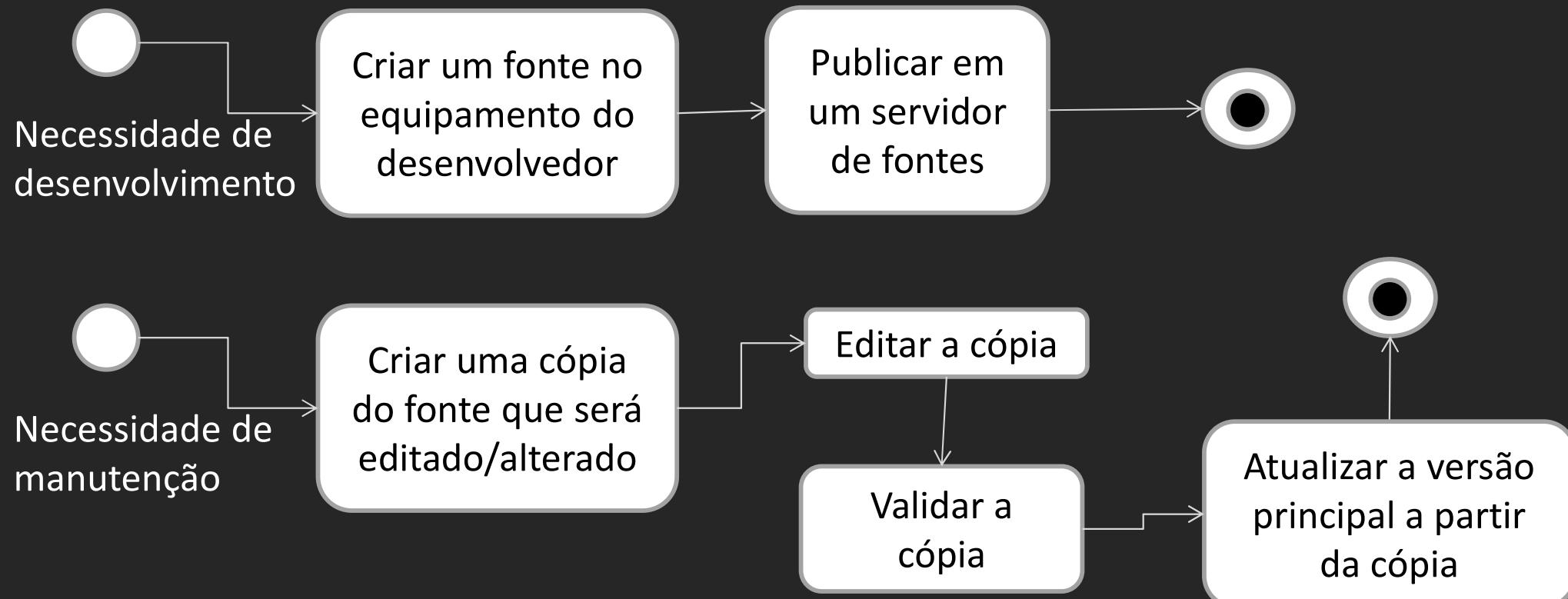


Para guardar os casos de testes, fontes de aplicação e demais documentos de desenho e gerenciamento de projeto, você escolheu criar um processo de controle de versões de documentos e fontes de projetos e selecionou a ferramenta (recurso) GIT para implementar essa mudança. A GD (Gerência de Desenvolvimento) da empresa de Dilan está sofrendo bastante com o controle de fontes de aplicação e de documentação de projeto e espera-se sanar essa questão. Hoje, cada grupo de projeto acaba definindo uma forma de trabalhar e não é incomum a sobreposição de fontes de forma errada, ou atualização de fonte em versão que não a última, gerando problemas em builds e releases.

Dilan gostaria que a sua empresa tivesse uma solução única para controlar fonte de programas e documentos, que permitisse a colaboração de desenvolvedores sobre um mesmo material, sem perder controle sobre a última versão oficial. Um processo foi desenhado por Consuelo e ela preparou um treinamento sobre GIT que você vai realizar agora!

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Vamos trabalhar com a seguinte proposta de processo de gestão de mudança no desenvolvimento de software:



ASSISTA OS VÍDEOS NO CANAL DO PROFESSOR



<https://youtu.be/t1E-cbB4gFY>

A screenshot of a Windows File Explorer window. The left sidebar shows a navigation tree with 'Área de Trabalho' expanded, showing 'Google Drive', 'Downloads', 'Documentos', 'OneDrive', 'Imagens', 'Material do prof', 'Materiais do prof', 'Média', and 'mp3recorder'. Below this is another 'Área de Trabalho' section with 'OneDrive', 'RunMIDIA', 'Este Computador', 'Área de Treball', 'Documentos', 'Downloads', 'Imagens', 'Músicas', and 'Vídeos'. The main pane displays a folder structure under 'RunMIDIA &gt; Google Drive'. It lists three items: 'Metodologia' (modified 02/02/2016 11:56, type folder), 'Docência (I)' (modified 14/03/2016 21:06, type folder), and 'Recibos 2016' (modified 11/07/2016 16:41, type folder). To the right of the file list is a video player window showing a man with a headset, likely the professor, speaking. The video player has a play button icon and a progress bar.

Conteúdo didático complementar - Controle de Versão de Documentos



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Em seguida, vamos vamos adotar uma solução na nuvem como ferramenta para administrar os fontes – o GIT e GIT HUB:



*Existem outras soluções, hoje menos populares que o GIT mas que funcionam bem na gestão de fontes e versões...*



# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Através da ferramenta de controle de versões de códigos e/ou documentos de software, é possível evitar os problemas de se trabalhar com a fonte errada em um determinado ponto do projeto, facilitando a colaboração no projeto.



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Trabalhando com GIT, os arquivos fontes serão organizados em:

### Cópia Master

Contém os arquivos na versão estável, que podem ser usados por outros desenvolvedores na integração de componentes ou com outros sistemas, ou podem ser usados para gerar um pacote de versão final do produto.

### Cópia Branch

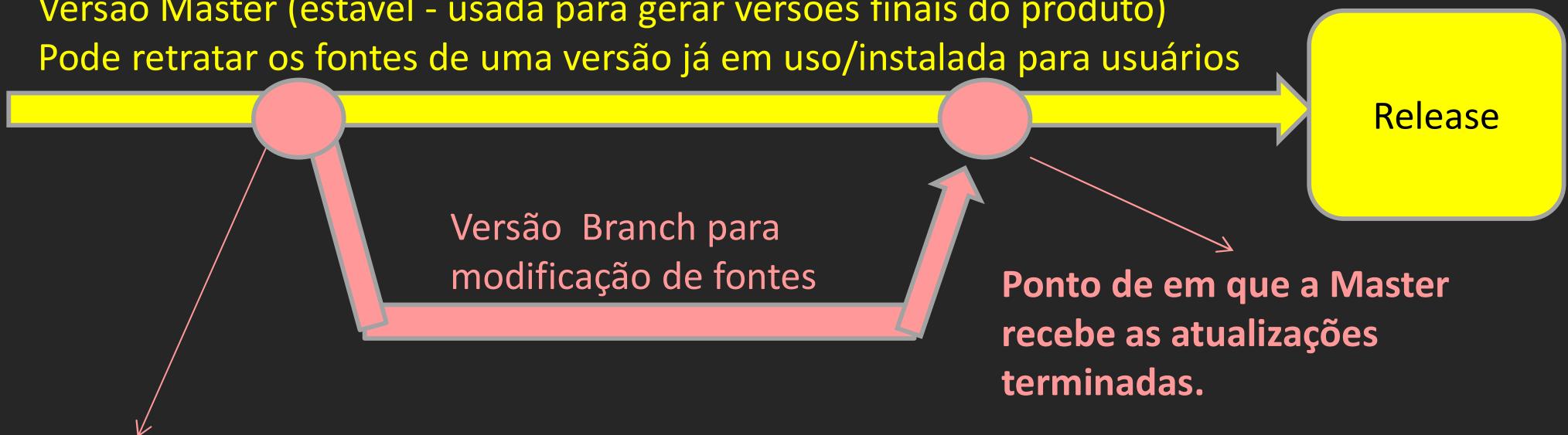
Contém os arquivos em uma versão de manutenção/atualização que não estão estáveis e não podem ser usados para gerar uma versão final do produto.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

O funcionamento da Branch.

Versão Master (estável - usada para gerar versões finais do produto)

Pode retratar os fontes de uma versão já em uso/instalada para usuários

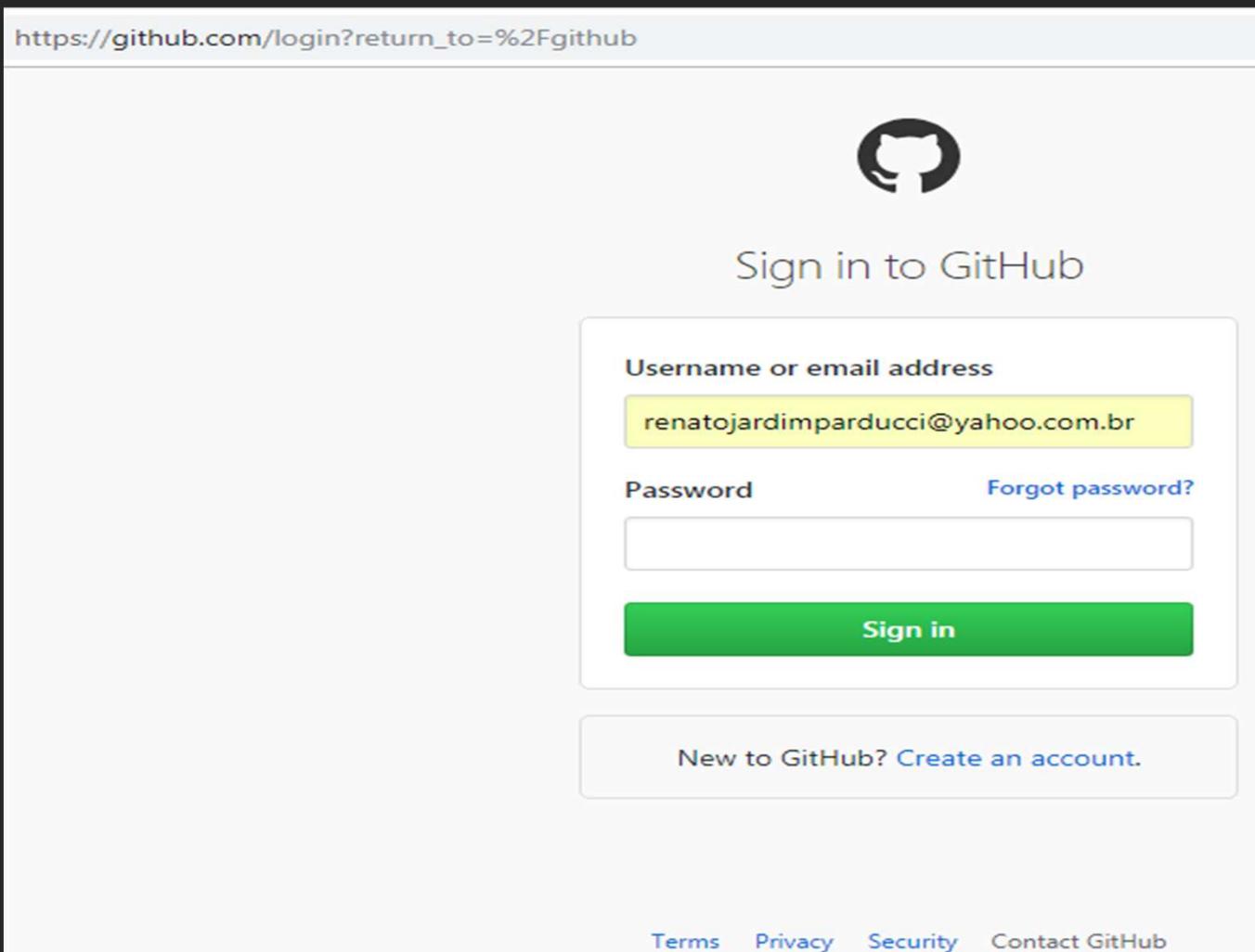


**Ponto de necessidade de manutenção para:**

- Corrigir o software/eliminar BUGs (manutenção CORRETIVA);
- Adaptar o software para novas regras de negócio (manutenção ADAPTATIVA);
- Prevenir contra possíveis problemas futuros (manutenção PREVENTIVA);
- Alcançar a perfeição na experiência do usuário (manutenção PREFECTIVA)

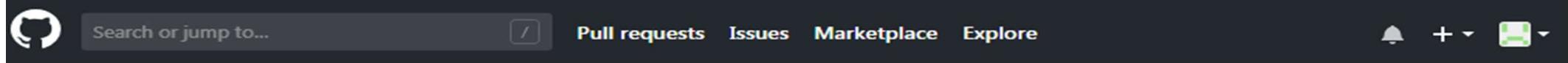
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Crie sua conta no GIT HUB.



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Faça o login para ver se está tudo Ok.



The screenshot shows a GitHub profile page with the following pinned repositories:

- fetch**: A window.fetch JavaScript polyfill. (JavaScript, 20.8k stars, 1.9k forks)
- hub**: hub helps you win at git. (Go, 14.1k stars, 1.4k forks)
- training-kit**: Open source on demand courses and cheat sheets for Git and GitHub. (HTML, 1.9k stars, 1.9k forks)
- choosealicense.com**: A site to provide non-judgmental guidance on choosing a license for your open source project. (Ruby, 1.4k stars, 457 forks)
- scientist**: A Ruby library for carefully refactoring critical paths. (Ruby, 4.7k stars, 200 forks)
- gh-ost**: GitHub's Online Schema Migrations for MySQL. (Go, 5.9k stars, 453 forks)

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Logado no GITHUB, acesse a área de repositórios.

The screenshot shows the GitHub user profile page for RenatoJardimParducci. The top navigation bar includes links for Search or jump to..., Pull requests, Issues, Marketplace, Explore, and a notifications icon. A large red arrow points from the text above down to the 'Your repositories' link in the dropdown menu on the right. The main content area displays the GitHub logo, the tagline "How people build software.", location information (San Francisco, CA), contact details (https://github.com/about and support@github.com), and a Verified badge. Below this, there are three pinned repository cards: 'fetch', 'hub', and 'training-kit'. The 'Your repositories' link is part of a vertical menu on the right side of the page.

Signed in as  
RenatoJardimParducci

Your profile

**Your repositories**

Your stars

Your gists

Help

Settings

Sign out

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Seus repositórios de projetos de software aparecerão, caso você já tenha catalogado algum.

The screenshot shows a GitHub profile interface. At the top, there is a pro tip message: "ProTip! Updating your profile with your name, location, and a profile picture helps other GitHub users get to know you." with a green "Edit profile" button and a close "X" icon. Below this, the user's statistics are displayed: Overview, **Repositories 12** (which is underlined in red), Stars 0, Followers 0, and Following 0. There is also a search bar with placeholder text "Find a repository...", and dropdown menus for "Type: All" and "Language: All", along with a green "New" button. The main content area lists three repositories: **1TDSJ** (updated on 26 Apr 2017), **RepExemplo** (updated on 17 Jan 2017), and **ExemploTDSS**.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Seus repositórios de projetos de software aparecerão, caso você já tenha catalogado algum.

The screenshot shows a GitHub profile interface. At the top, there is a pro tip message: "ProTip! Updating your profile with your name, location, and a profile picture helps other GitHub users get to know you." with a green "Edit profile" button and a close "X" icon. Below this, the user's statistics are displayed: Overview, **Repositories 12** (which is underlined in red), Stars 0, Followers 0, and Following 0. There is also a search bar with placeholder text "Find a repository...", and dropdown menus for "Type: All" and "Language: All", along with a green "New" button. The main content area lists three repositories: **1TDSJ** (updated on 26 Apr 2017), **RepExemplo** (updated on 17 Jan 2017), and **ExemploTDSS**.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Crie um repositório.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner RenatoJardimParducci / Repository name TesteGITHUB ✓

Great repository names are short and memorable. Need inspiration? How about sturdy-spoon.

Description (optional)

 Public Anyone can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with a README This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None ⓘ

Create repository

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Entenda a tela do GitHub.

The screenshot shows a GitHub repository page. At the top, it displays the repository name 'RenatoJardimParducci / TesteGITHUB'. Below the name, there are buttons for 'Unwatch' (with 1 watch), 'Star' (0 stars), and 'Fork' (0 forks). A navigation bar includes links for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. A note below the navigation bar states 'No description, website, or topics provided.' with an 'Edit' button. Under the repository details, there are sections for 'Branch: master' (1 commit), '1 branch' (1 branch), '0 releases' (0 releases), and '1 contributor' (1 contributor). Below these, there are buttons for 'Create new file', 'Upload files' (highlighted with a red box and arrow), 'Find file', and 'Clone or download'. A commit history is shown with one entry: 'RenatoJardimParducci Initial commit' (a minute ago). The commit details show 'README.md' (Initial commit, a minute ago). A red box highlights the 'README.md' link, and a red arrow labeled '2' points to it from below. Another red box highlights the 'Upload files' button, and a red arrow labeled '3' points to it from above. The repository name 'TesteGITHUB' is visible at the bottom left.

- 1 Seleciona a área/cópia de fontes para trabalho
- 2 Nomes dos arquivos que constam na área
- 3 Usado para carregar arquivos fonte no GIT

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Você pode criar pastas para separar tipos de arquivos

The screenshot shows a GitHub repository page for 'RenatoJardimParducci / TesteGITHUB'. The page includes navigation tabs for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. A red callout box with the text 'Crie um nome para a pasta/nome de um arquivo Readme que será criado' has an arrow pointing to the 'Create new file' button in the main repository actions area. Below this, there are buttons for Upload files, Find file, and Clone or download. The repository details show 1 commit, 1 branch, 0 releases, and 1 contributor. The latest commit was made a minute ago by RenatoJardimParducci. A file named README.md is listed with an initial commit a minute ago. The repository name 'TesteGITHUB' is displayed at the bottom.

RenatoJardimParducci / TesteGITHUB

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided.

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request

Create new file Upload files Find file Clone or download

RenatoJardimParducci Initial commit Latest commit 7296d6c a minute ago

README.md Initial commit a minute ago

README.md

TesteGITHUB

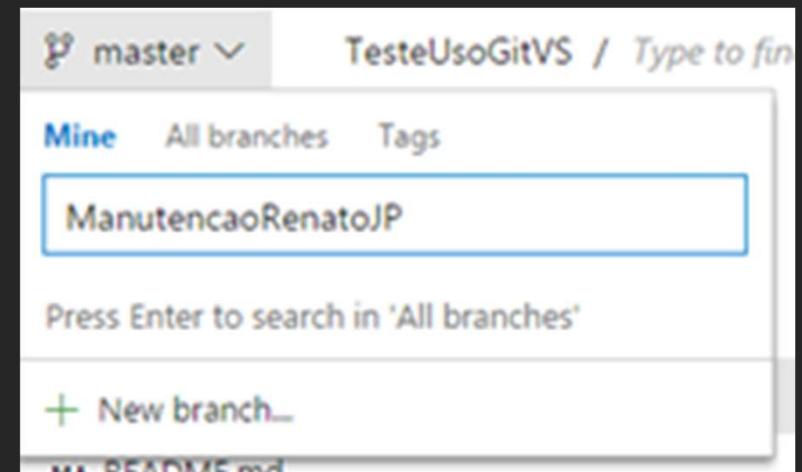
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Para incluir novos fontes ou alterar fontes publicados no GIT, crie uma Branch (cópia de desvio) da área Master.

A Master deve conter apenas os códigos fonte estáveis, que podem ser usados por outros desenvolvedores.

A Branch é uma réplica da Master para que um ou mais programadores façam alterações nos programas e depois republiquem os arquivos atualizados na Master.

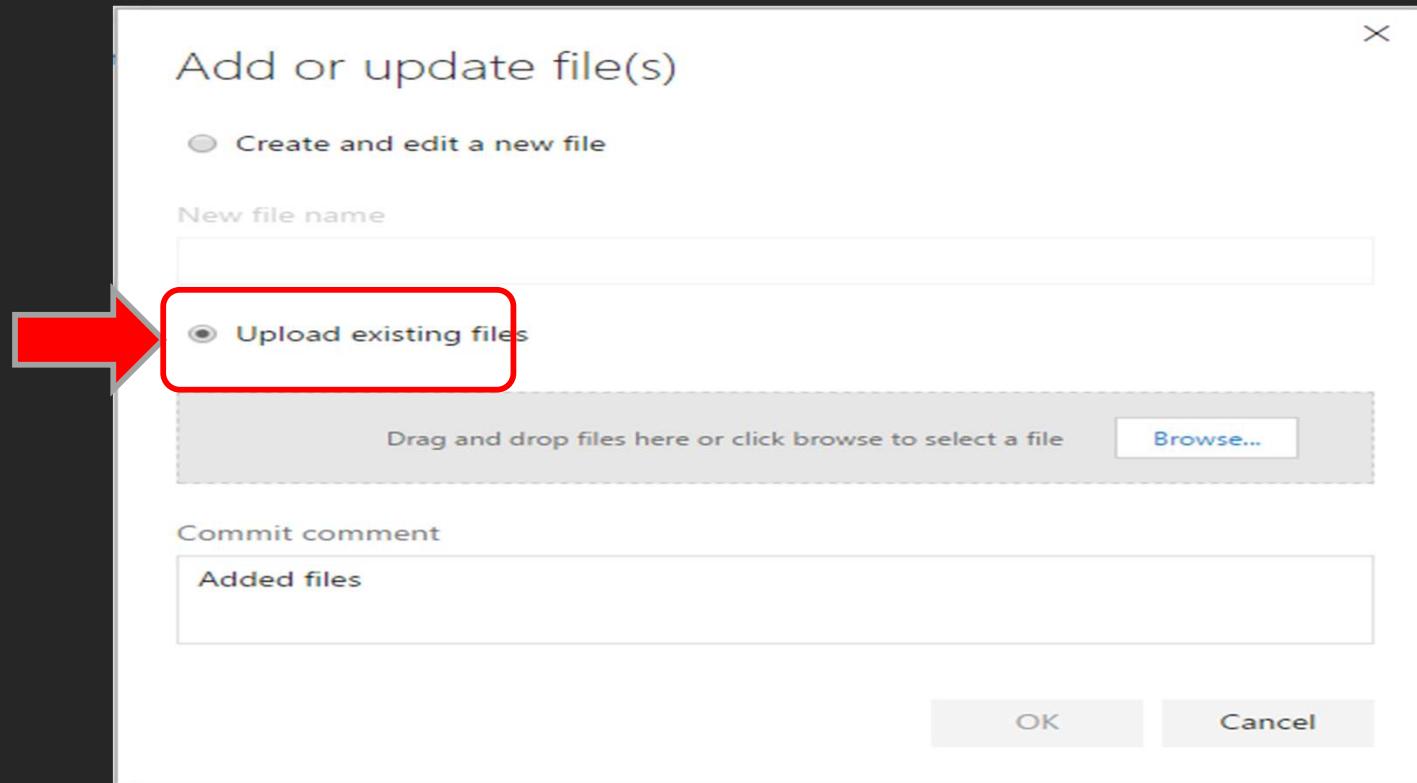
Crie uma Branch a partir da Master com o nome que desejar.



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Acesse o link de Upload para subir para o GitHub um arquivo do seu computador.

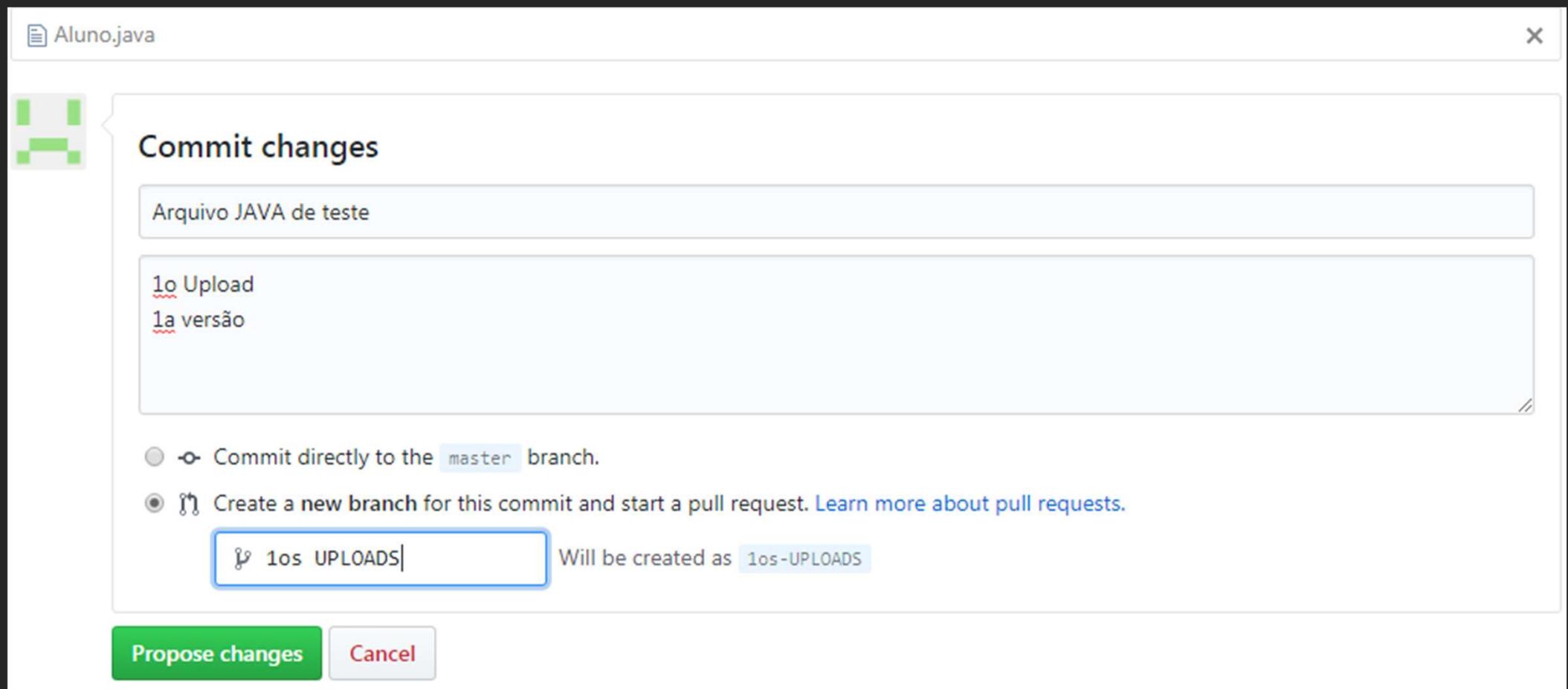
Suba um arquivo .JAVA ou .SQL para experimentar!



Como alternativa, você pode abrir a pasta com o seu arquivo no Windows Explorer e arrastá-lo para a página do GITHUB.

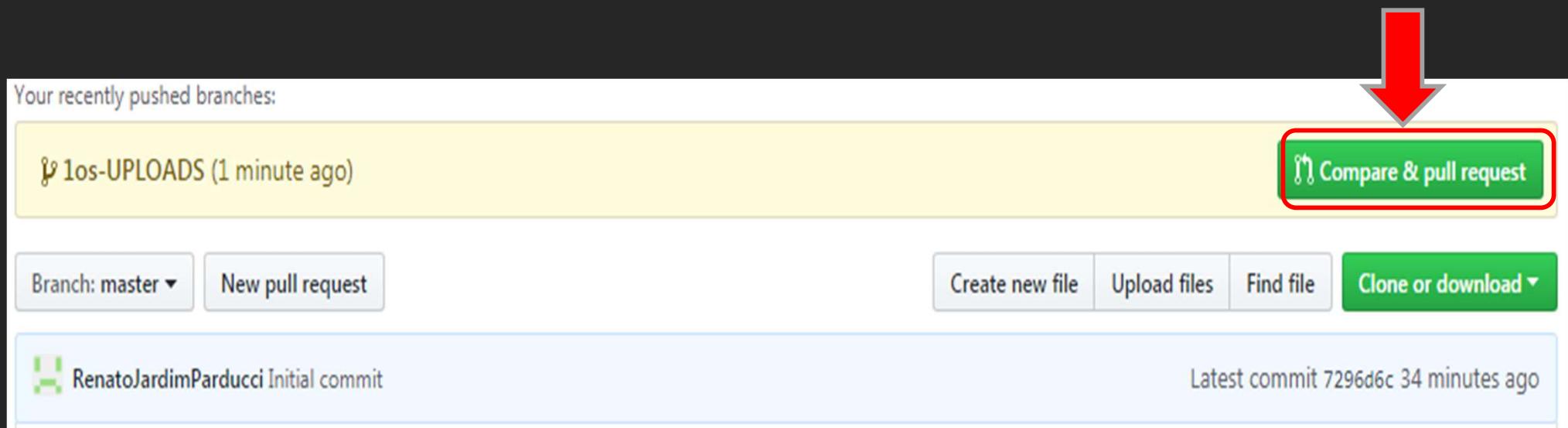
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Cada inclusão ou alteração de arquivo pode ser comentada ao ser salva, facilitando a interpretação das versões.!



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Quando algo é modificado na área de trabalho/cópia, fica habilitada a possibilidade de criar uma Pull request para atualizar a Master a partir da Branch.!



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Depois que estiver certo que a mudança está completa e correta, publique a modificação na cópia Master, tornando-a disponível para todos os desenvolvedores usarem!

The screenshot shows a pull request interface for a repository named 'TesteUsoGitVS'. The tab 'Pull Requests' is selected. A specific pull request is displayed, titled 'Updated CalcHoras.js'. The 'Description' section contains the note '- Updated CalcHoras.js'. Below the description, it says 'Markdown supported. Use # to mention a work item or @ to mention a person.' and lists a bullet point: '• Updated CalcHoras.js'. The 'Reviewers' section shows '[ProjetoExemplo-2TBA-2016]\ProjetoExemplo-2TBA-2016 Team' and a search bar 'Search users and groups'. The 'Work Items' section has a search bar 'Search work items by ID or title'. At the bottom, there is a red arrow pointing to a blue button labeled 'New pull request'.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Experimente criar uma nova Branch para fazer alterações.!

Faça as modificações e execute o Commit.

Crie uma Pull request e execute a atualização da Master (Merge).

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

A Master terá o histórico de todas as modificações feitas.

The screenshot shows a GitHub repository interface. At the top, there are navigation links: Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. Below this, the branch is set to 'master' and the file is 'TesteGITHUB / Aluno.java'. On the right, there are 'Find file' and 'Copy path' buttons. A red arrow points to the 'History' button in the bottom right corner of the commit card. The commit card for 'e92ad94' shows it was made 9 minutes ago by 'RenatoJardimParducci' with the message 'Update Aluno.java'. It also indicates '1 contributor'. Below the commit card, the file content is displayed:

```
1 public class Aluno1 extends Pessoa {  
2  
3     private Matricula matricula;  
4  
5     public void SolicitarMatricula() {  
6  
7         }  
8  
9 }
```

The 'History' button is highlighted with a red box and a red arrow pointing to it from the top right.

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

A Master terá o histórico de todas as modificações feitas.

The screenshot shows a GitHub repository page for 'RenatoJardimParducci / TesteGITHUB'. The top navigation bar includes 'Unwatch' (1), 'Star' (0), 'Fork' (0), and tabs for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. A dropdown menu indicates the current branch is 'master'. The main content area displays a list of commits made on November 30, 2018:

- Merge pull request #1 from RenatoJardimParducci/1os-UPLOADS ...  
RenatoJardimParducci committed 10 minutes ago  
Verified | ba7d3c0 | ↗
- Update Aluno.java  
RenatoJardimParducci committed 14 minutes ago  
Verified | e92ad94 | ↗
- Arquivo JAVA de teste ...  
RenatoJardimParducci committed 23 minutes ago  
Verified | 55ecc9c | ↗
- Initial commit  
RenatoJardimParducci committed an hour ago  
Verified | 7296d6c | ↗

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Caso você precise recuperar uma versão anterior, basta selecioná-la.

The screenshot shows a GitHub repository page for 'RenatoJardimParducci / TesteGITHUB'. The 'Code' tab is selected. A dropdown menu shows the branch is 'master'. Below the header, there are links for Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. The main content area displays a list of commits:

- Merge pull request #1 from RenatoJardimParducci/1os-UPLOADS ...  
RenatoJardimParducci committed 9 minutes ago
- Update Aluno.java  
RenatoJardimParducci committed 12 minutes ago
- Arquivo JAVA de teste ...  
RenatoJardimParducci committed 21 minutes ago
- Initial commit  
RenatoJardimParducci committed 44 minutes ago

At the bottom of the commit list, a tooltip says 'Browse the repository at this point in the history'. The browser's address bar shows the URL: <https://github.com/RenatoJardimParducci/TesteGITHUB/tree/55ecc9c3ecf99d5fc8bf61652825d9ca3a2da336>. The browser's taskbar shows several open windows: SourceTreeSetup....exe, Git-2.19.2-32-bit (2).exe, and another Git-2.19.2-32-bit.exe window. The system tray shows the date and time as 11:43 30/11/2018.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Quando selecionada uma versão do fonte, o GIT mostra o que foi alterado para você ter certeza de qual versão está vendo.

The screenshot shows a GitHub commit history for the file `Aluno.java`. The commit was made by RenatoJardimParducci 12 minutes ago and is verified. It has one parent commit `55ecc9c` and a commit hash `e92ad942ca6b8d544ad4282d95dfcde8d1121567`. The interface shows 1 changed file with 1 addition and 1 deletion. The diff view highlights the change where the class name was updated from `Aluno` to `Aluno1`.

```
diff --git a/Aluno.java b/Aluno.java
@@ -1,4 +1,4 @@
- public class Aluno extends Pessoa {
+ public class Aluno1 extends Pessoa {
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Se precisar deletar uma Branch que ficou pra trás (não deletada no momento do MERGE), acesse na página principal do projeto, o número de branches e na janela que aparecer, clique na lata de lixo para eliminar a branch correspondente.

The screenshot shows a GitHub repository page for 'RenatoJardimParducci / TesteGITHUB'. The top navigation bar includes 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', and 'Wiki'. Below the navigation, it says 'No description, website, or topics provided.' and has a 'Manage topics' link. On the left, there's a sidebar with '8 commits' and a 'Branch: master' dropdown set to 'master'. A 'New pull request' button is also visible. The main content area shows a summary: '2 branches' (highlighted with a red box and a red arrow pointing to the trash icon in the delete dialog). Below this, there's a 'Default branch' section for 'master' (updated a minute ago by RenatoJardimParducci) and a 'Your branches' section listing 'SUPERBRANCH-01' (updated 2 minutes ago by RenatoJardimParducci). At the bottom, there's an 'Active branches' section also listing 'SUPERBRANCH-01'.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Você acabou de estudar um processo com atividades, definição de responsabilidades e ferramentas para gerenciar fontes de programas de aplicação em suas versões.

Somado com o conhecimento de JUNIT você deu o 1º passo para garantir a Qualidade em projetos de software e a Governança.

ASSISTA OS VÍDEOS NO CANAL DO PROFESSOR



<https://youtu.be/MYhIM0bk9aQ>

A screenshot of a video player displaying a GitHub tutorial page. The page features a large image of a man's face on the right. The video player interface includes a play button, volume control, and a progress bar showing 0:30 / 13:49. A tooltip on the screen says "Visualize your project's community". Below the video player, the text "Conteúdo didático complementar - GitHub" is visible.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Read the guide Start a project

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you follow.

Visualize your project's community

A new graph is available in the Graphs tab to visualize your repository's data. You can now explore how repositories that contain Ruby gems relate to other repositories on GitHub.

View 7 new broadcasts

Your repositories CC BY-NC-ND

YouTube

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Você pode ainda, baixar o aplicativo GIT para Windows ou outra plataforma em <https://git-scm.com/download/win>, para fazer comandos da sua máquina.

### Downloading Git

The screenshot shows the 'Your download is starting...' screen of the Git for Windows download page. A large red arrow points to the '32-bit Git for Windows Setup' link, which is highlighted with a red border. Below it, other download options are listed: 'Git for Windows Setup' and '64-bit Git for Windows Setup.' To the right, a 'Now What?' section provides links to 'Read the Book', 'Download a GUI', and 'Get Involved'.

Your download is starting...

You are downloading the latest (2.19.2) 32-bit version of **Git for Windows**. This is the most recent [maintained build](#). It was released **9 days ago**, on 2018-11-21.

If your download hasn't started, [click here to download manually](#).

**Other Git for Windows downloads**

[Git for Windows Setup](#)

**32-bit Git for Windows Setup**

[64-bit Git for Windows Setup.](#)

**Now What?**

Now that you have downloaded Git, it's time to start using it...

**Read the Book**

Dive into the Pro Git book and learn at your own pace.

**Download a GUI**

Several free and commercial GUI tools are available for the Windows platform.

**Get Involved**

A knowledgeable Git community is available to answer your questions.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Pode também selecionar um gerenciador de interface gráfica (Client – GIT) como o SourceTree para não ter que executar comandos GIT via linha de comando.

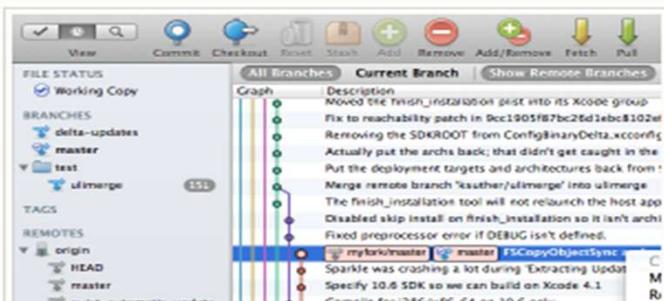
### GUI Clients

Git comes with built-in GUI tools for committing ([git-gui](#)) and browsing ([gitk](#)), but there are several third-party tools for users looking for platform-specific experience.

If you want to add another GUI tool to this list, just follow the [instructions](#).

[All](#)[Windows](#)[Mac](#)[Linux](#)[Android](#)[iOS](#)

22 Windows GUIs are shown below ↓

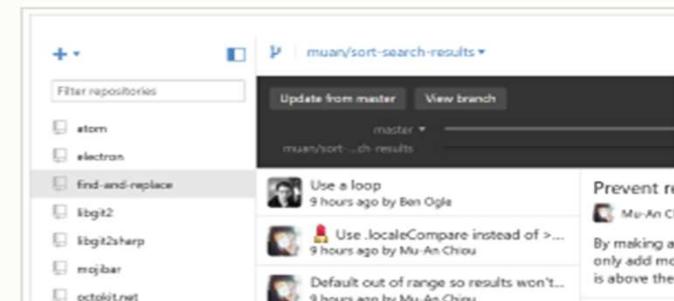


#### SourceTree

Platforms: Mac, Windows

Price: Free

License: Proprietary



#### GitHub Desktop

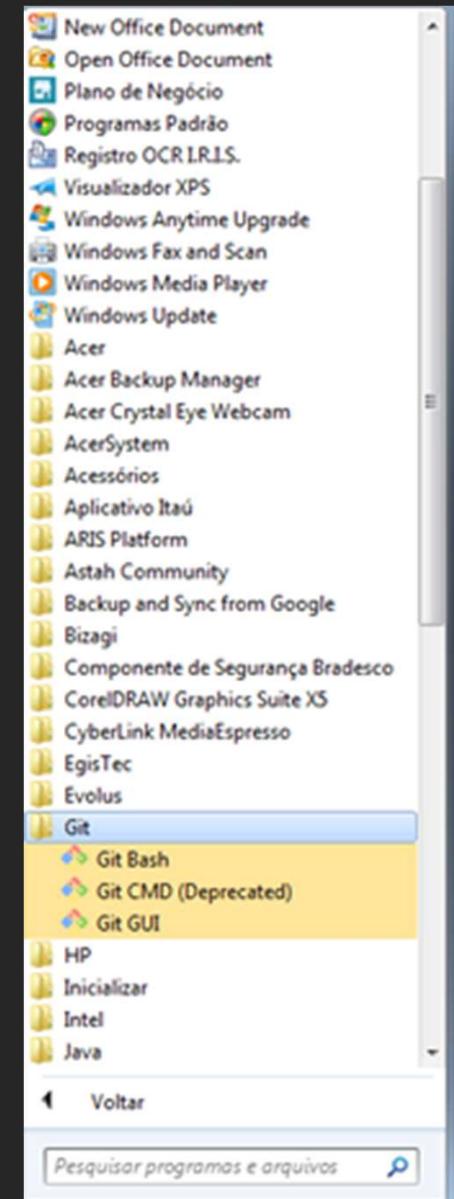
Platforms: Mac, Windows

Price: Free

License: MIT

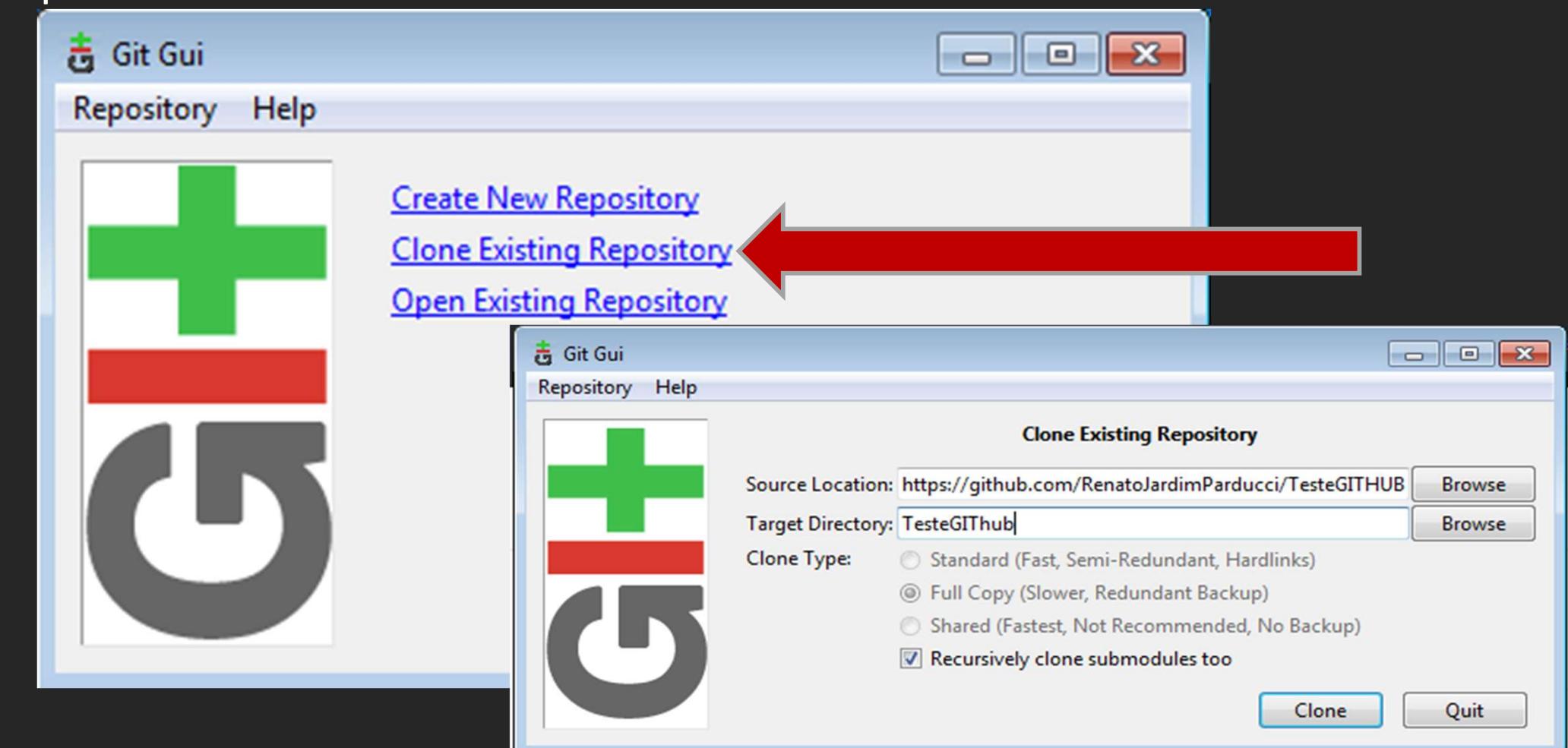
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Esses aplicativos adicionais permitem que você publique e puxe arquivos para manutenção usando somente o seu computador pessoal, sem necessidade de usar o navegador, como nós fizemos no nosso exemplo.



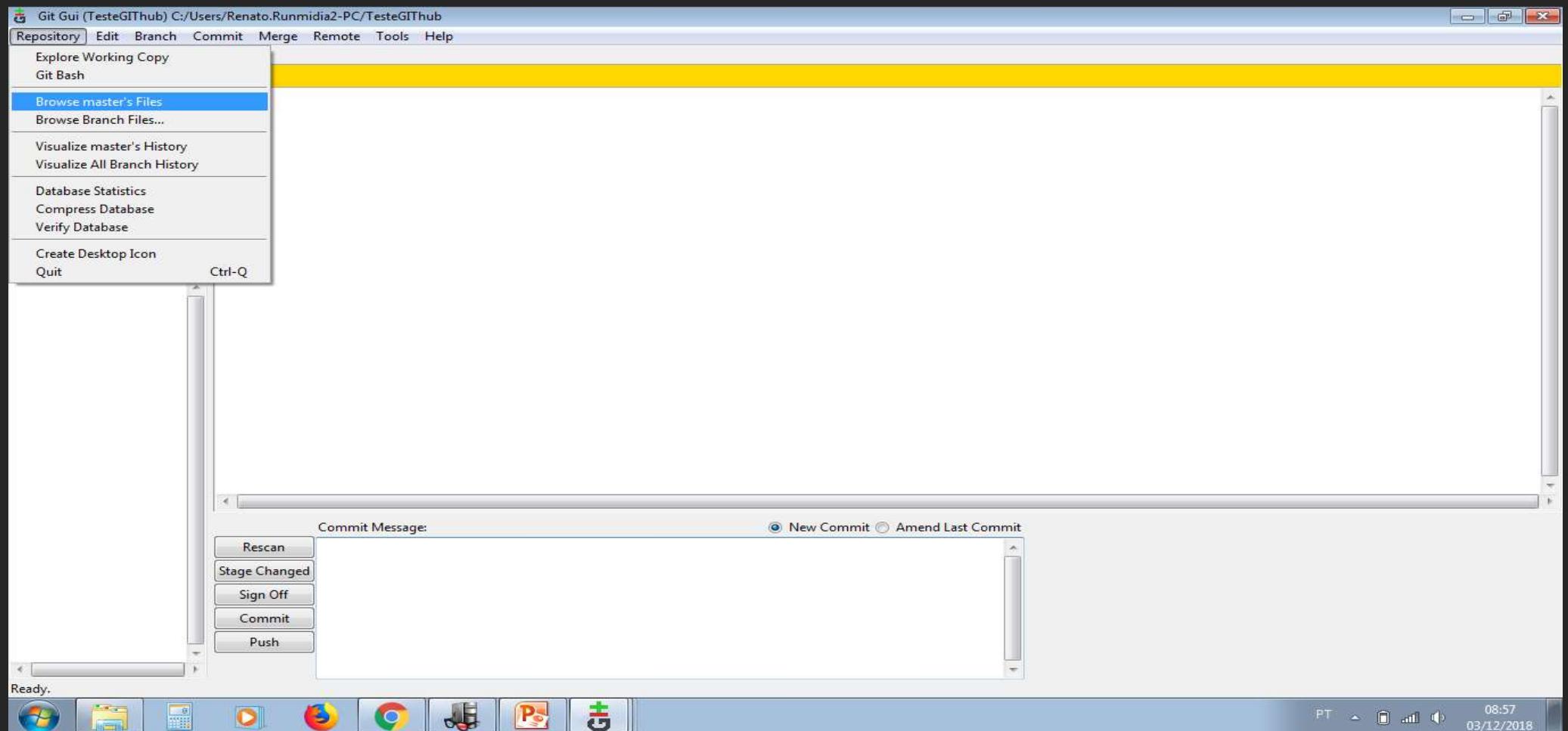
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Você pode sincronizar os repositórios em nuvem com o computador pessoal.



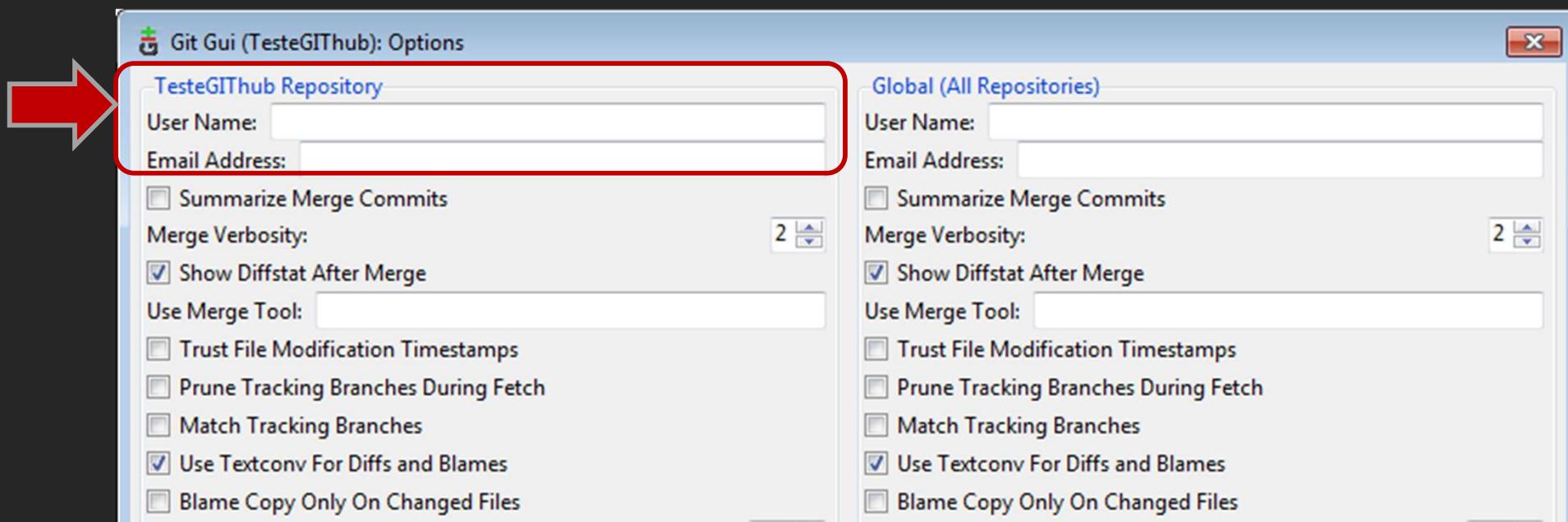
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Com o GitGUI, por exemplo, você pode acessar as áreas Master, Branches e pastas pelos menus o seu PC..



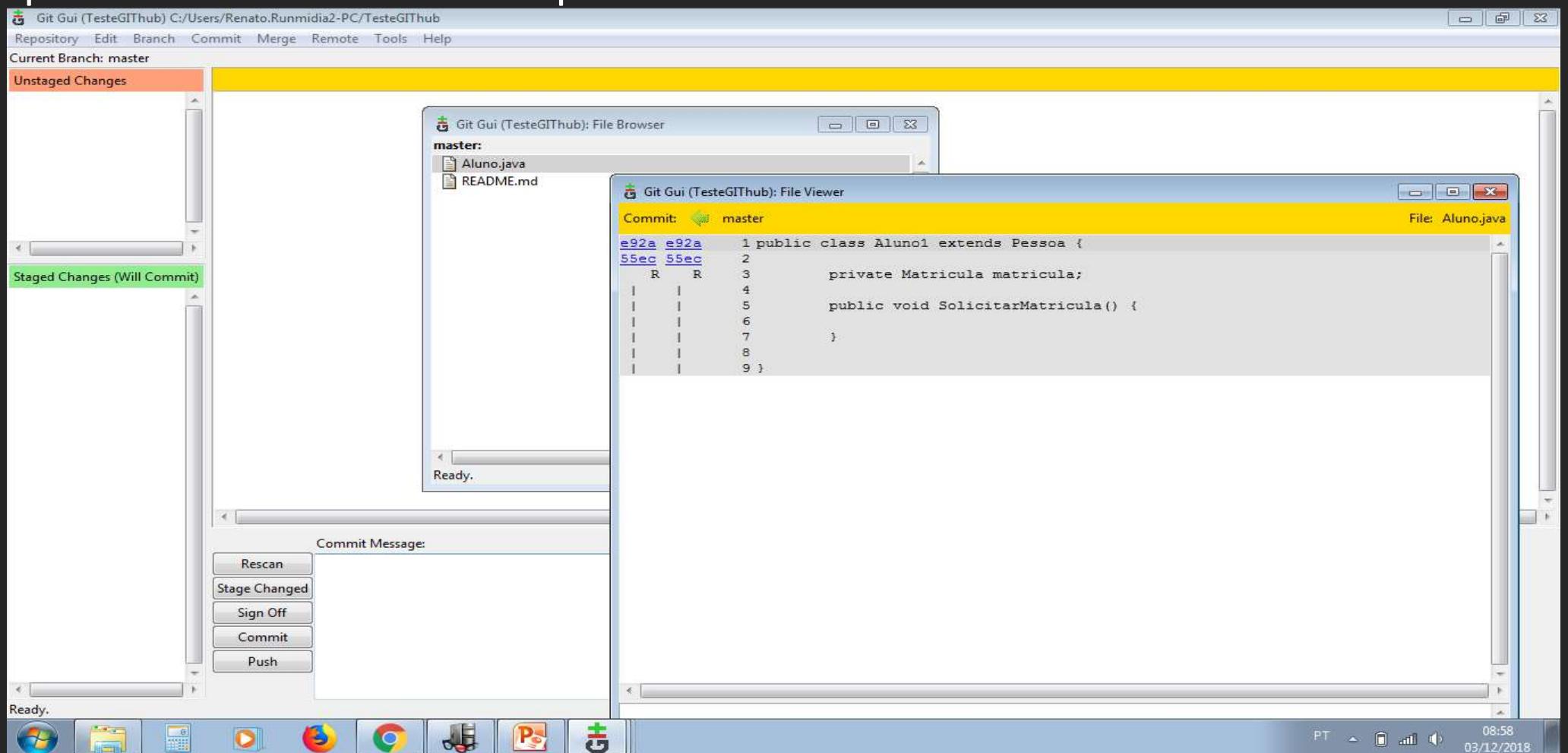
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Ele pode sincronizar com o GitHub.com se você configurar o Submenu Edit/Options com os dados do seu login na WEB.



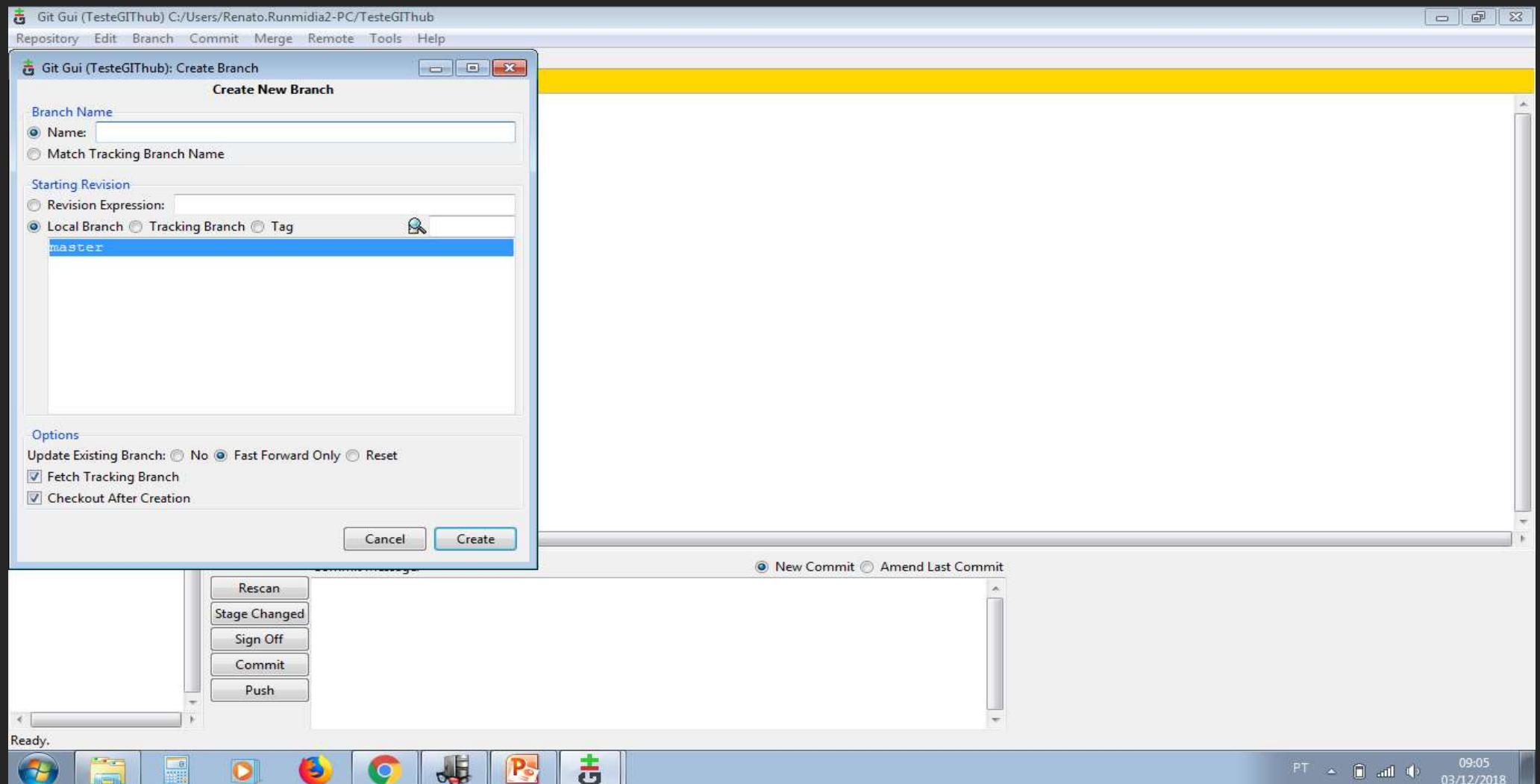
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Você pode sincronizar os repositórios em nuvem com o computador pessoal e visualizar seus repositórios.



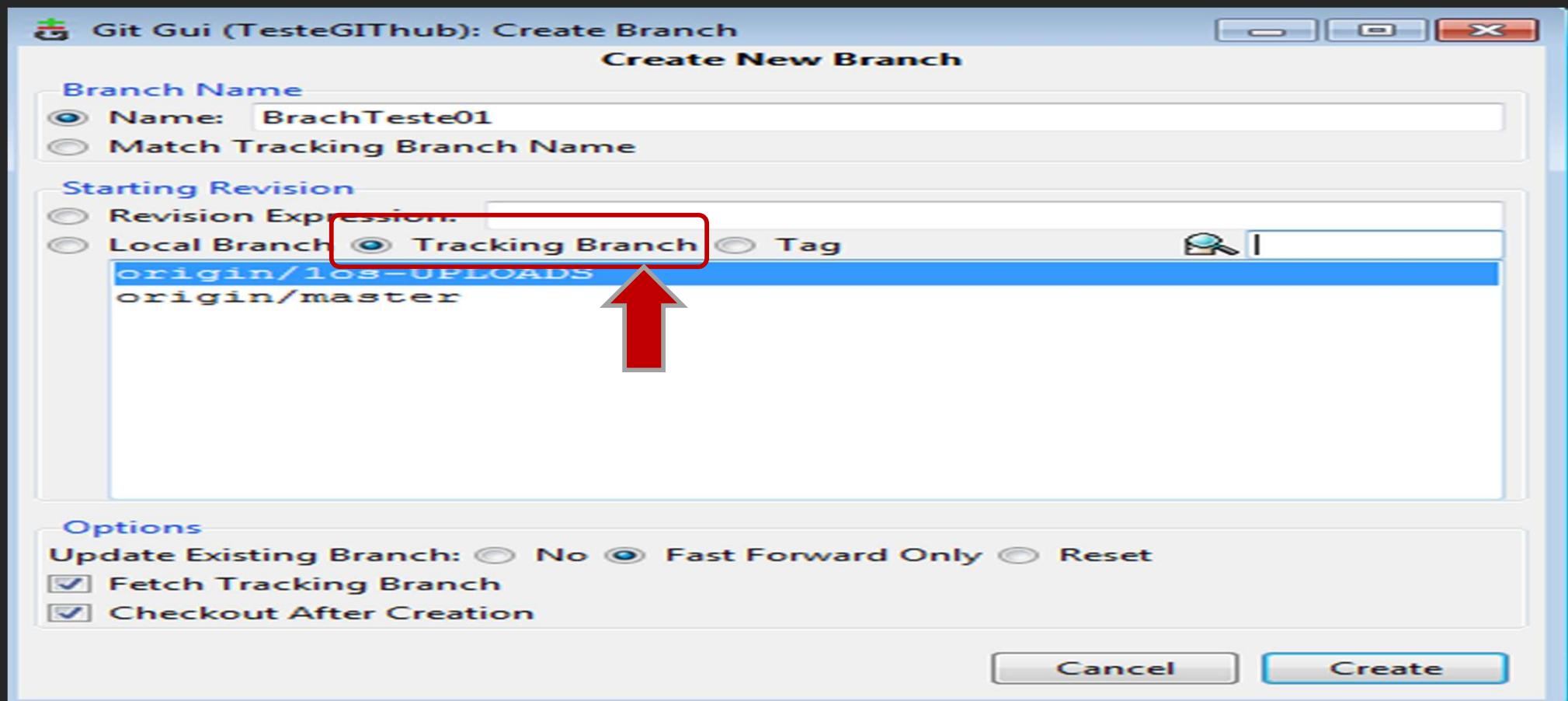
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Pode ainda criar branch...



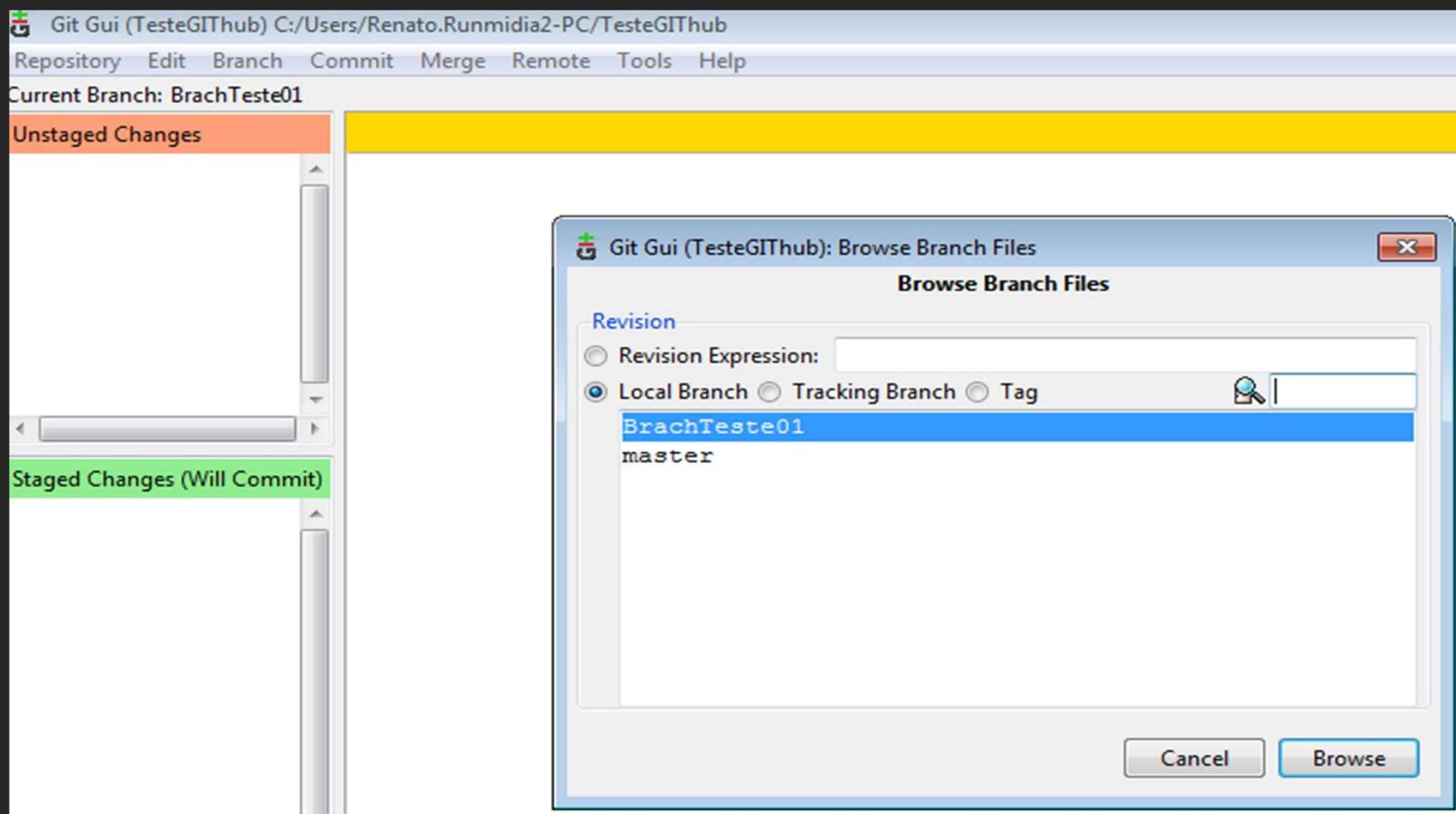
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

E essa Branch pode ser local (só no seu computador) ou rastreada (track), ligada com o repositório remoto na Internet (associada ao GitHub).



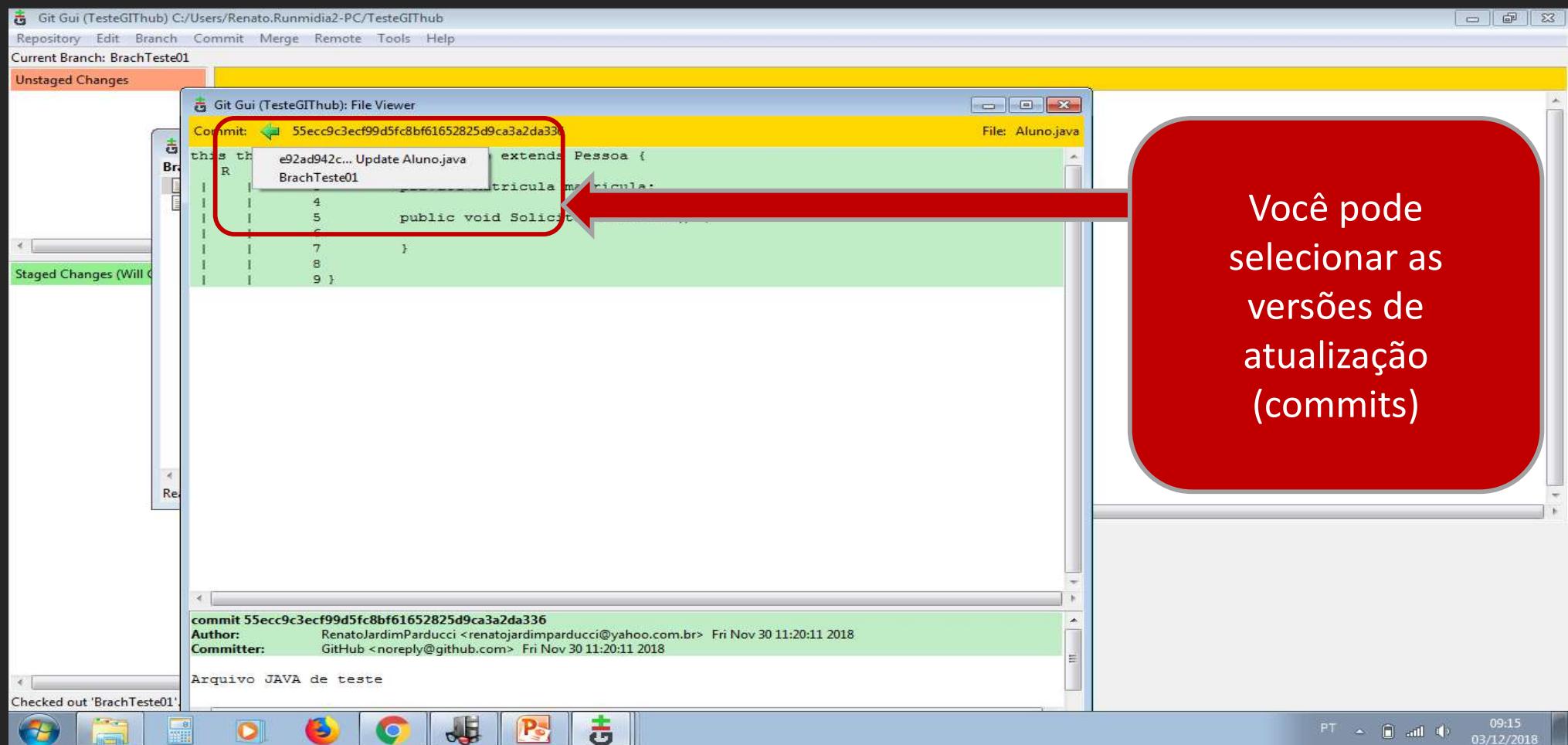
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Após criada a Branch, a navegação de acesso aos fontes é via menu da aplicação GUI.



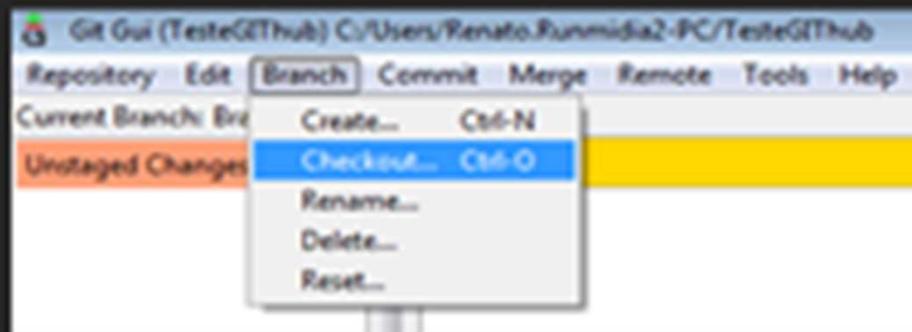
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Após criada a Branch, a navegação de acesso aos fontes é via menu da aplicação GUI.



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

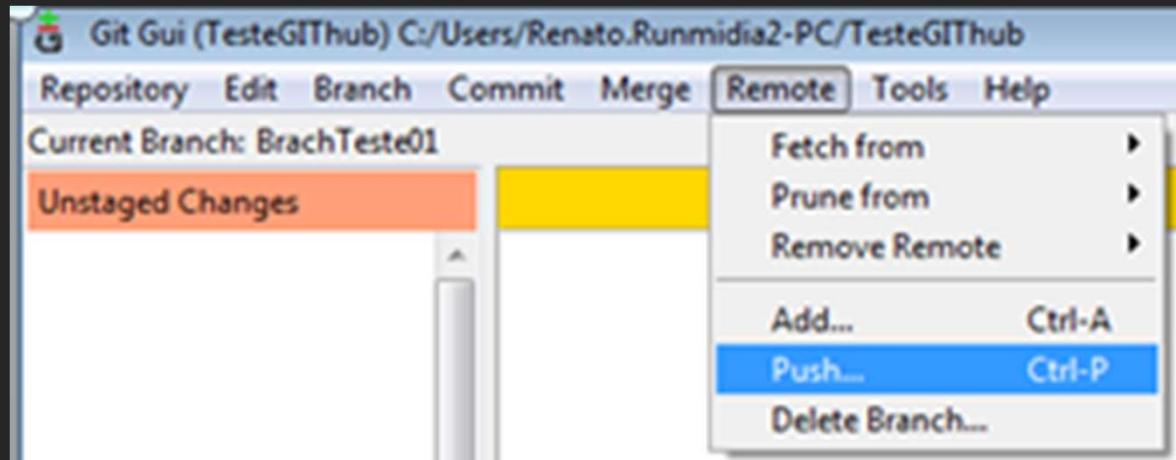
Para editar arquivos na Branch, faça um Checkout!



Use a opção REPOSITORY/EXPLORE WORKING COPY para acessar os documentos no repositório local e editá-los!

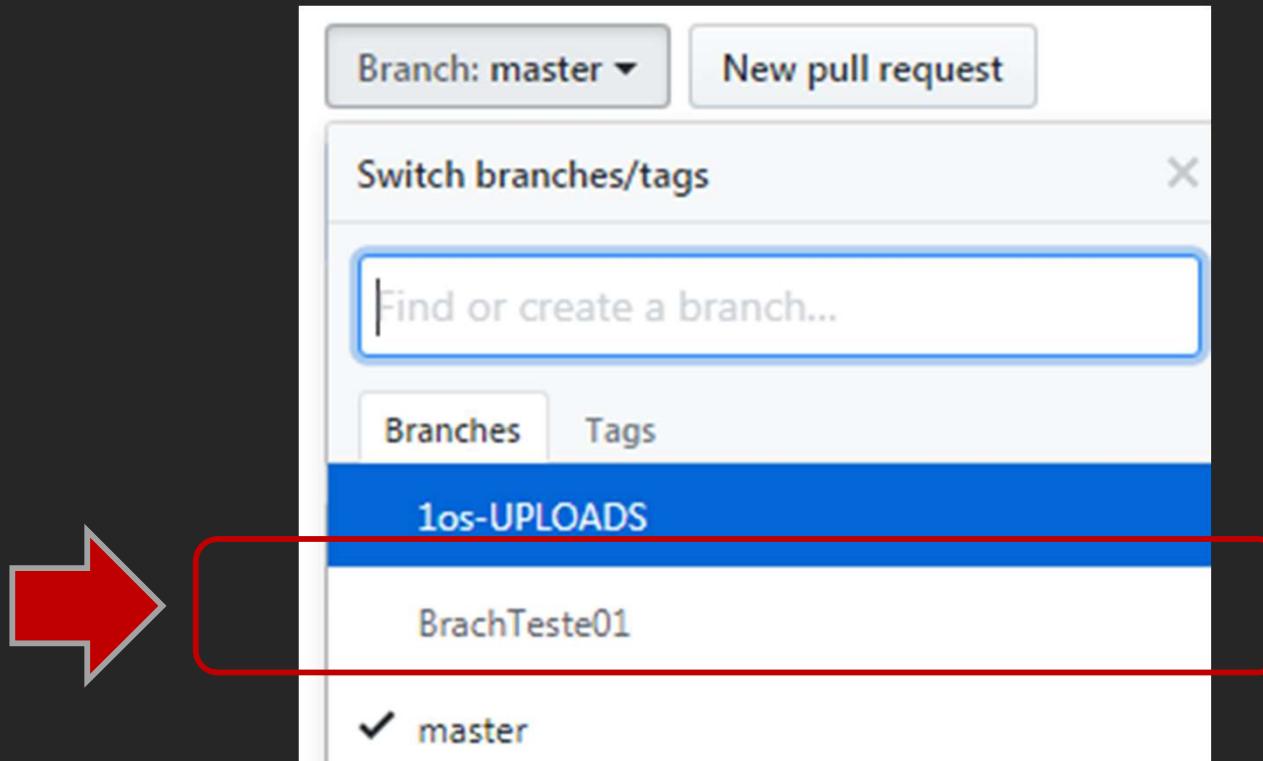
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Se você desejar publicar a Branch criada no seu PC no GitHub.com, faça um PUSH no submenu Remote (será exigido o Login do GitHub para essa operação)!



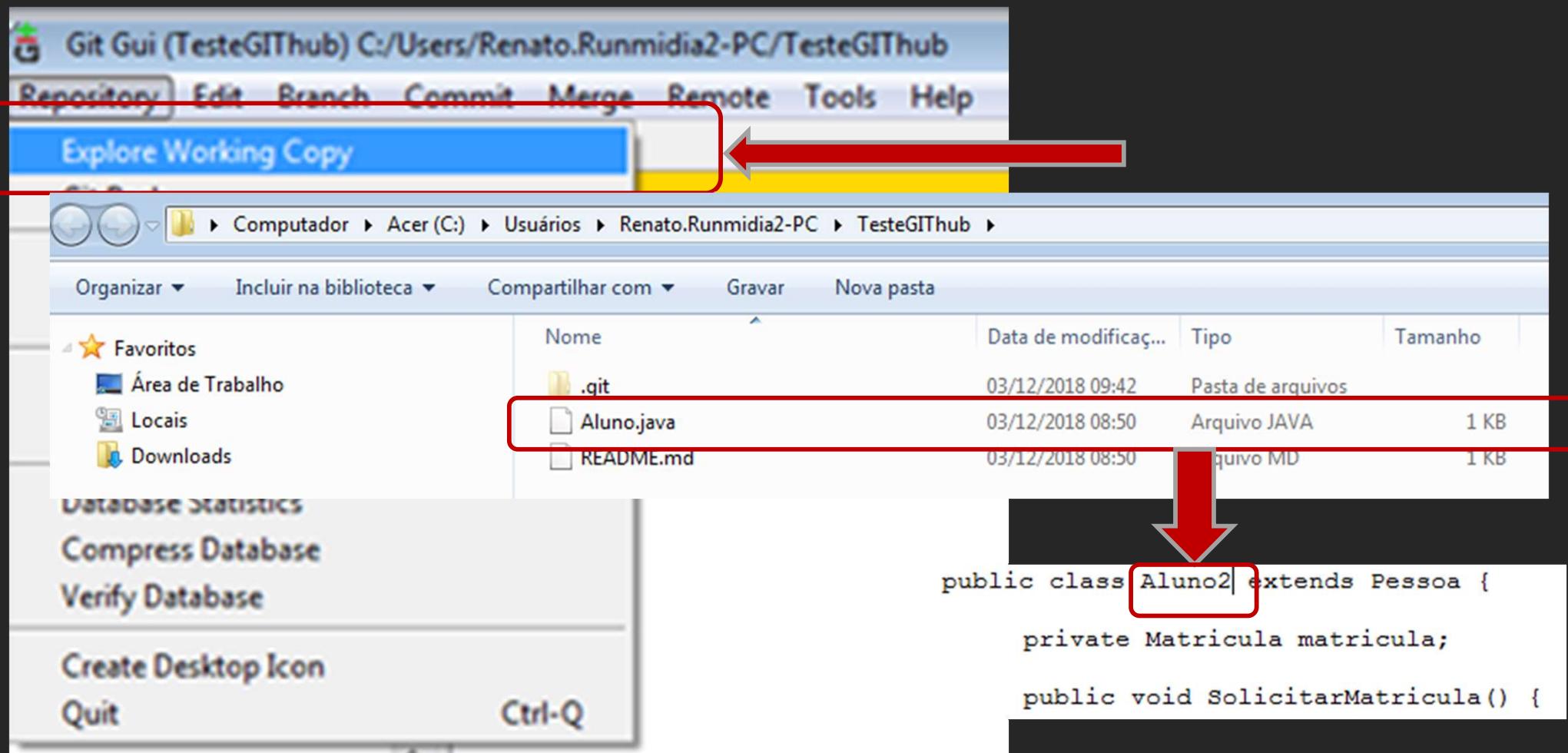
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Após o Push, você poderá visualizar e operar a Branch e seus arquivos na WEB:



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

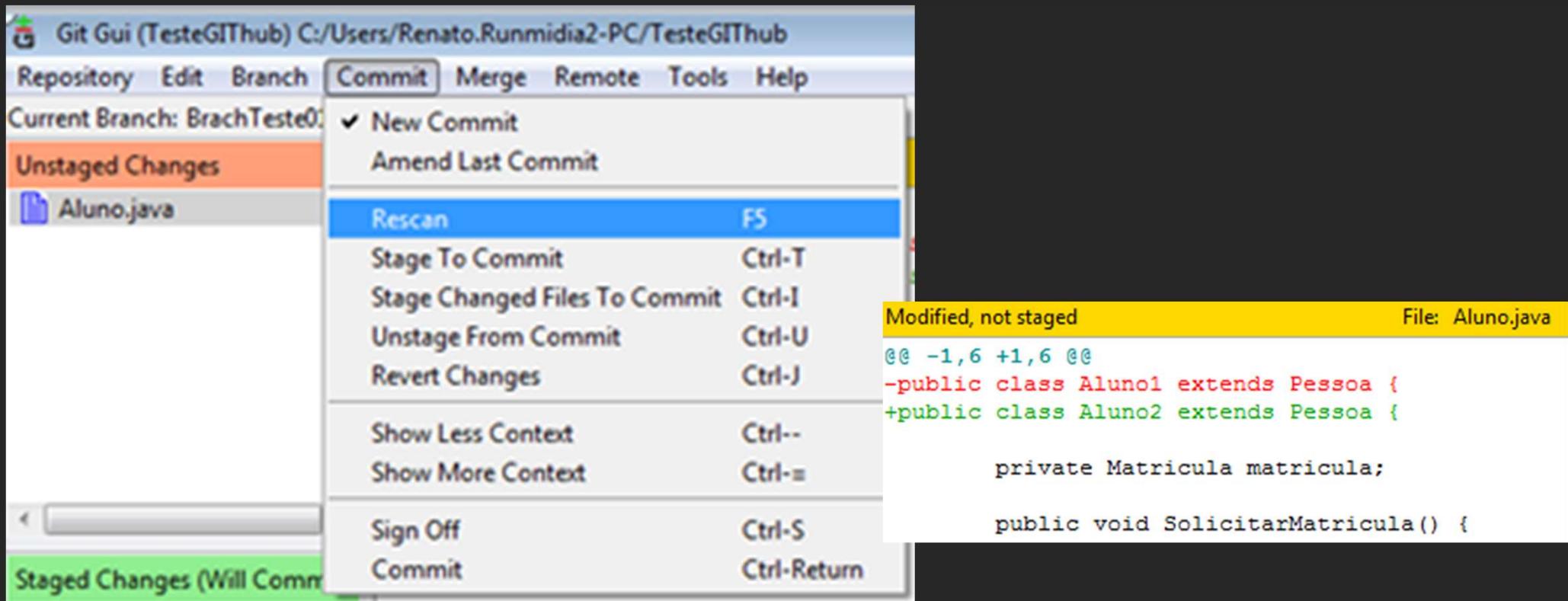
Você pode editar os arquivos no seu PC:



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

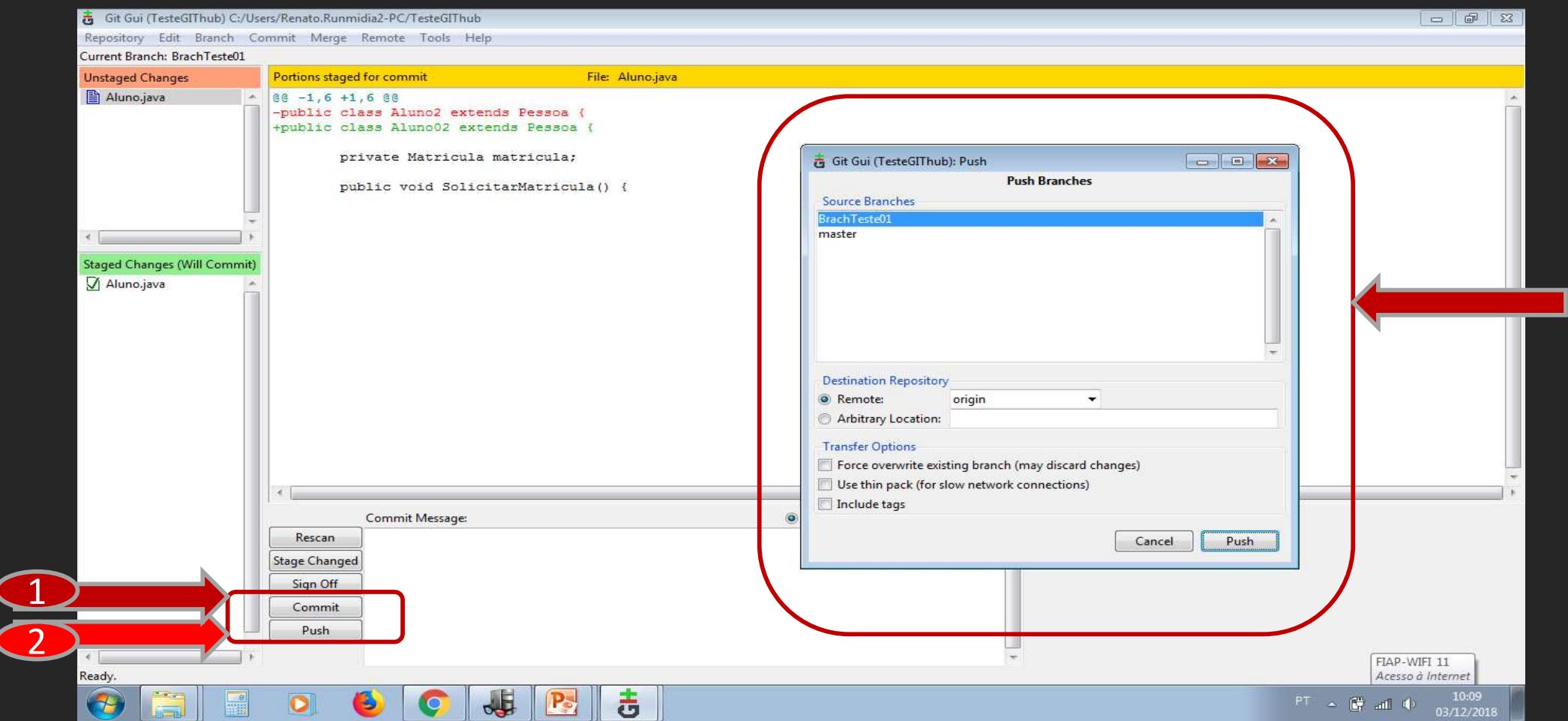
As edições são refletidas no GitGui e podem ser atualizadas por Push no GitHub!

Vá no submenu Commit/Rescan para descobrir as mudanças feitas, antes de fazer o Push!



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Executando o Push via Menu Commit: execute o Commit e depois o Push!!



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

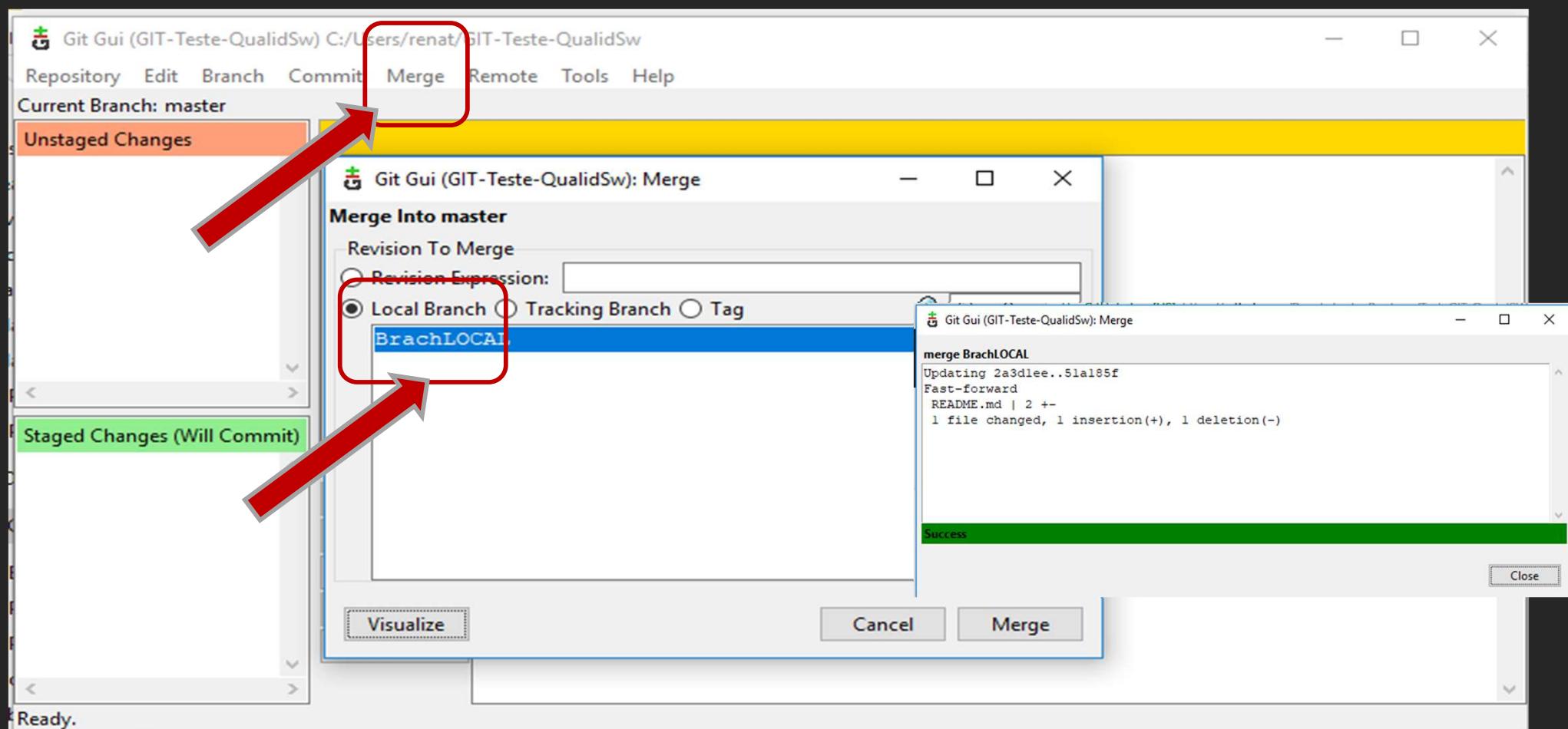
Com esses procedimentos, você consegue fazer Check out e Check in (Push) , mantendo sincronia entre o ambiente remoto dos fontes e o seu ambiente local de trabalho.

Após os Pushes, confira sempre se o GitHub.com foi atualizado corretamente.

The screenshot shows a GitHub repository page for 'RenatoJardimParducci / TesteGITHUB'. The top navigation bar includes 'Search or jump to...', 'Pull requests', 'Issues', 'Marketplace', 'Explore', and notification icons. The repository name is displayed above the main content area. Below the header, there's a section for 'Your recently pushed branches:' with a yellow highlight around 'BrachTeste01 (less than a minute ago)'. A green button labeled 'Compare & pull request' is visible next to it. At the bottom of the page, there are buttons for 'Branch: BrachTeste01', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The main content area shows a commit history with one entry from 'RenatoJardimParducci' titled 'Mudança via Cliente GUI' with a timestamp of '5 minutes ago'.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

**IMPORTANTE:** o fato de você ter atualizado a Branch e Master no GITHUB não implica que a Master Local tenha sido atualizada.  
Para atualizar a Master Local com as modificações, use o menu MERGE.



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

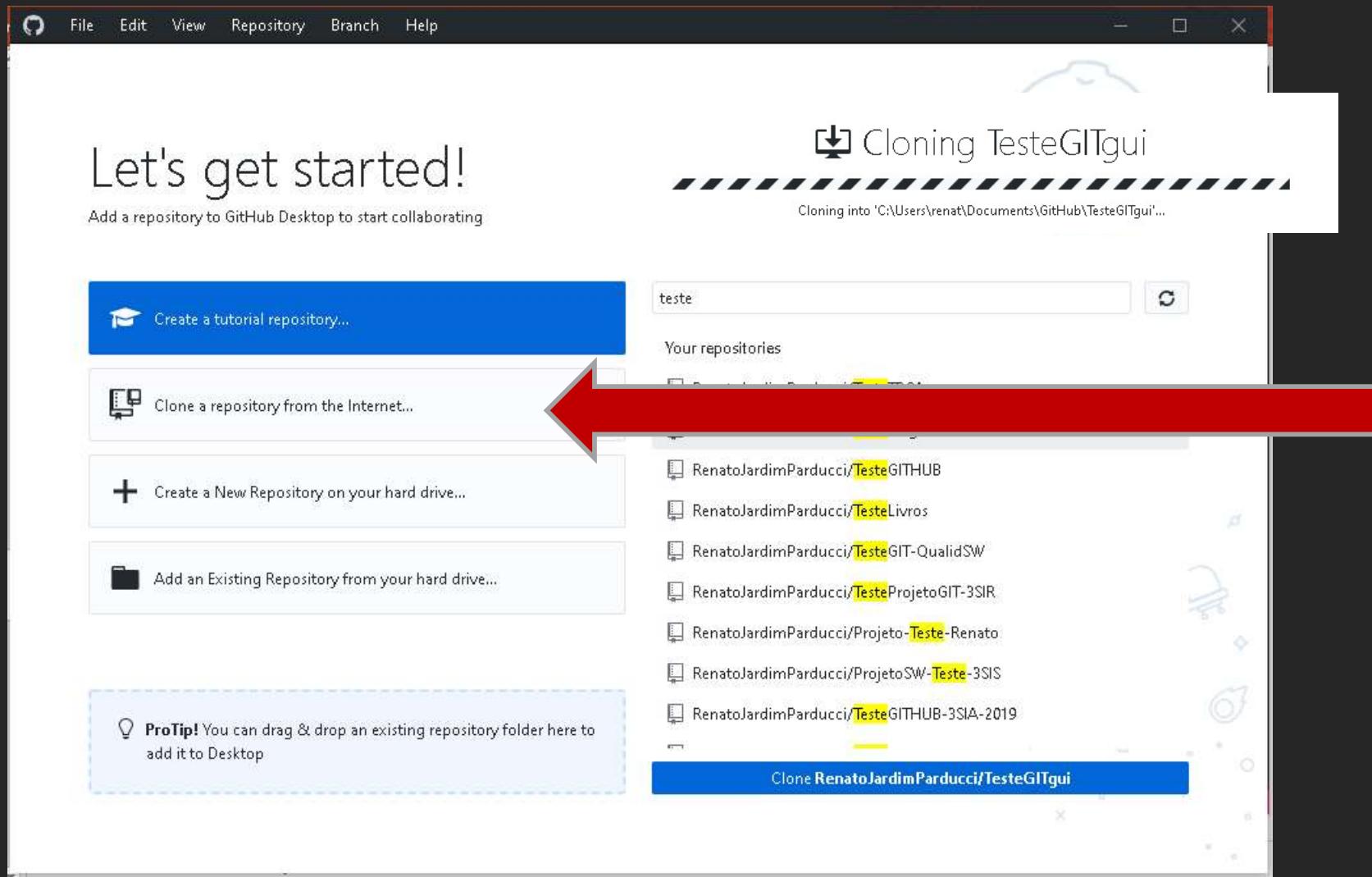
Esses aplicativos client permitem que você publique e puxe arquivos para manutenção usando somente o seu computador pessoal, sem necessidade de usar o navegador, como nós fizemos no nosso exemplo.



Vamos explorar agora o GitHub Desktop que pode ser obtido no próprio portal do GITHUB.

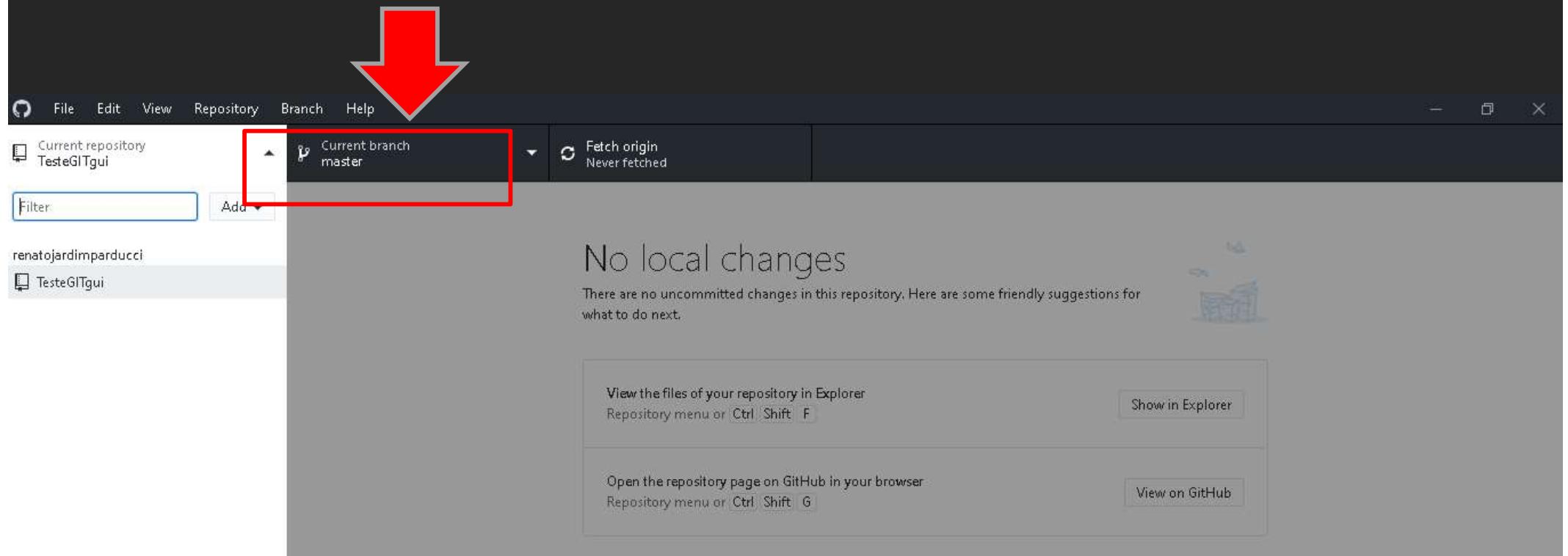
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Você pode sincronizar os repositórios em nuvem com o computador pessoal.



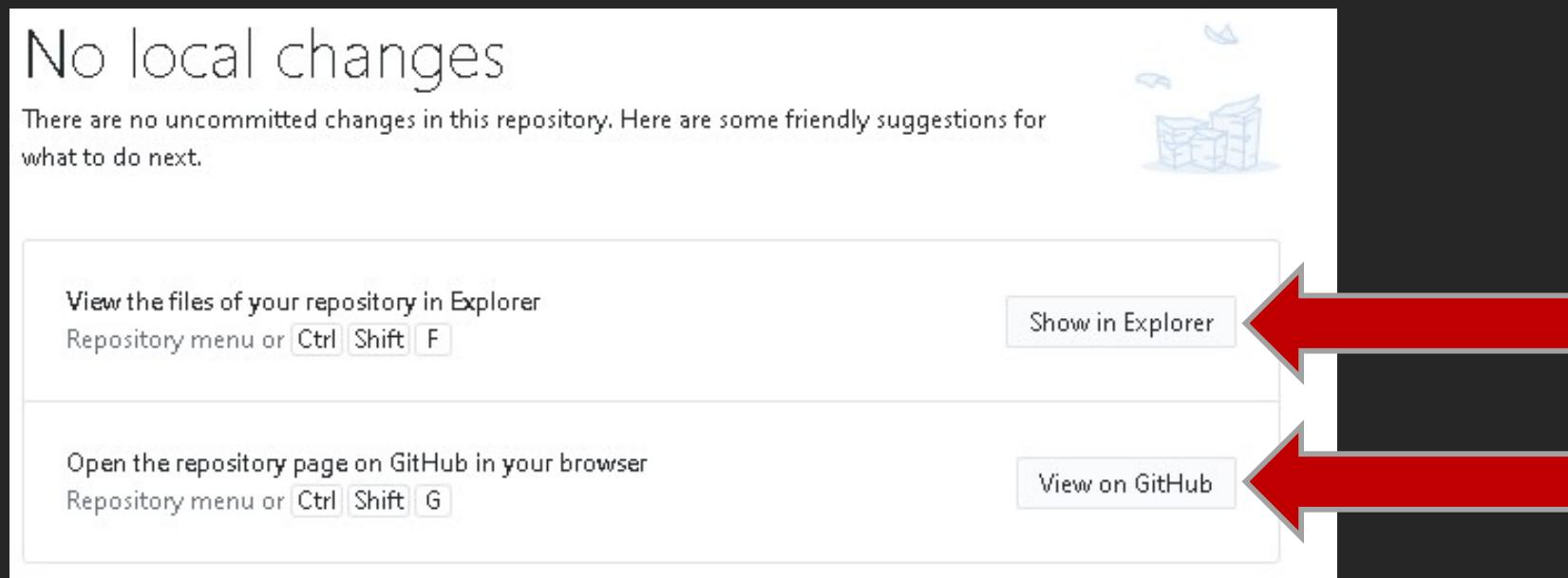
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Com o GitGUI, por exemplo, você pode acessar as áreas Master, Branches e pastas pelos menus o seu PC..



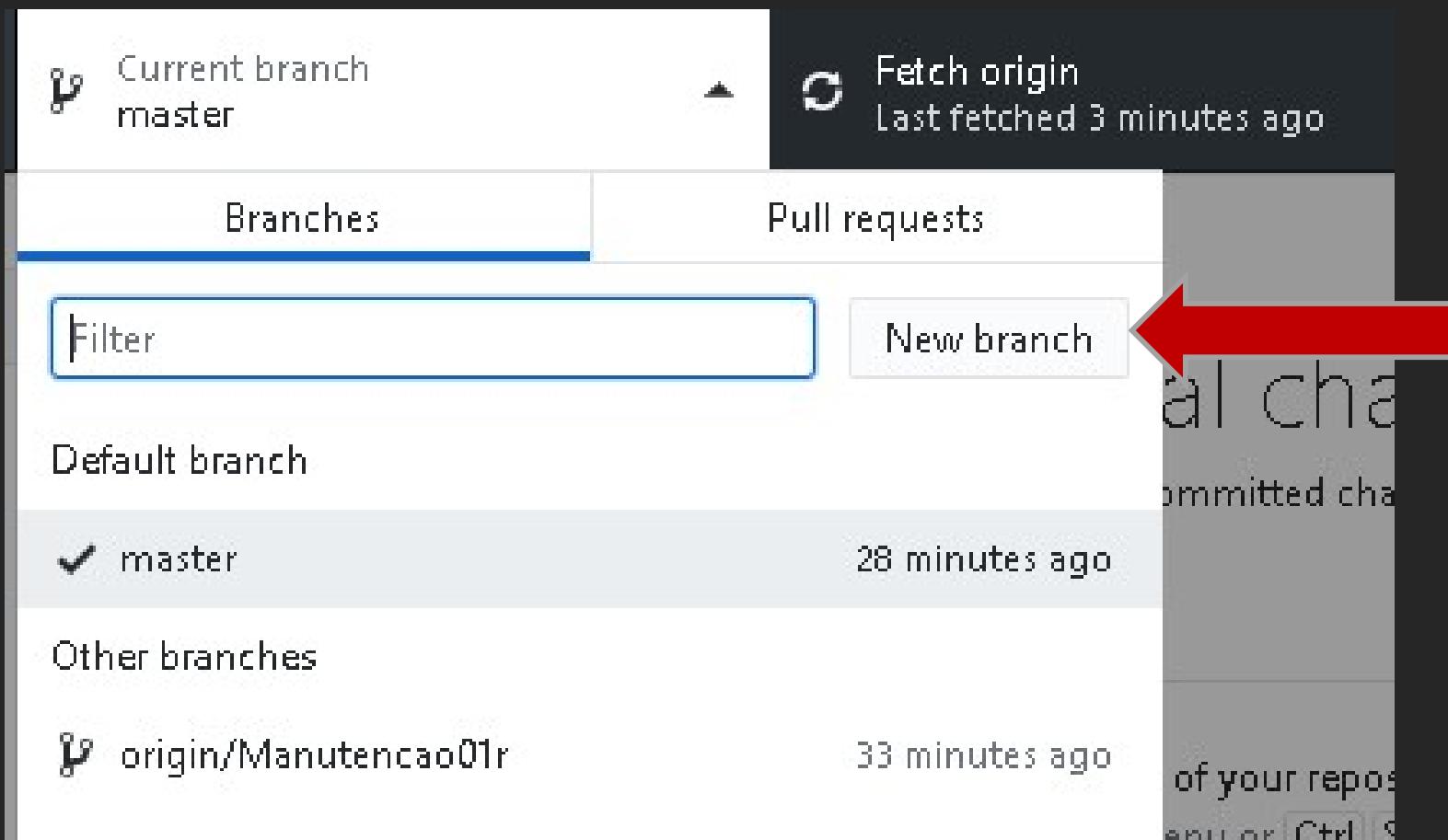
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Você pode acessar o clone local ou a pasta remota do GITHUB por esse software cliente.



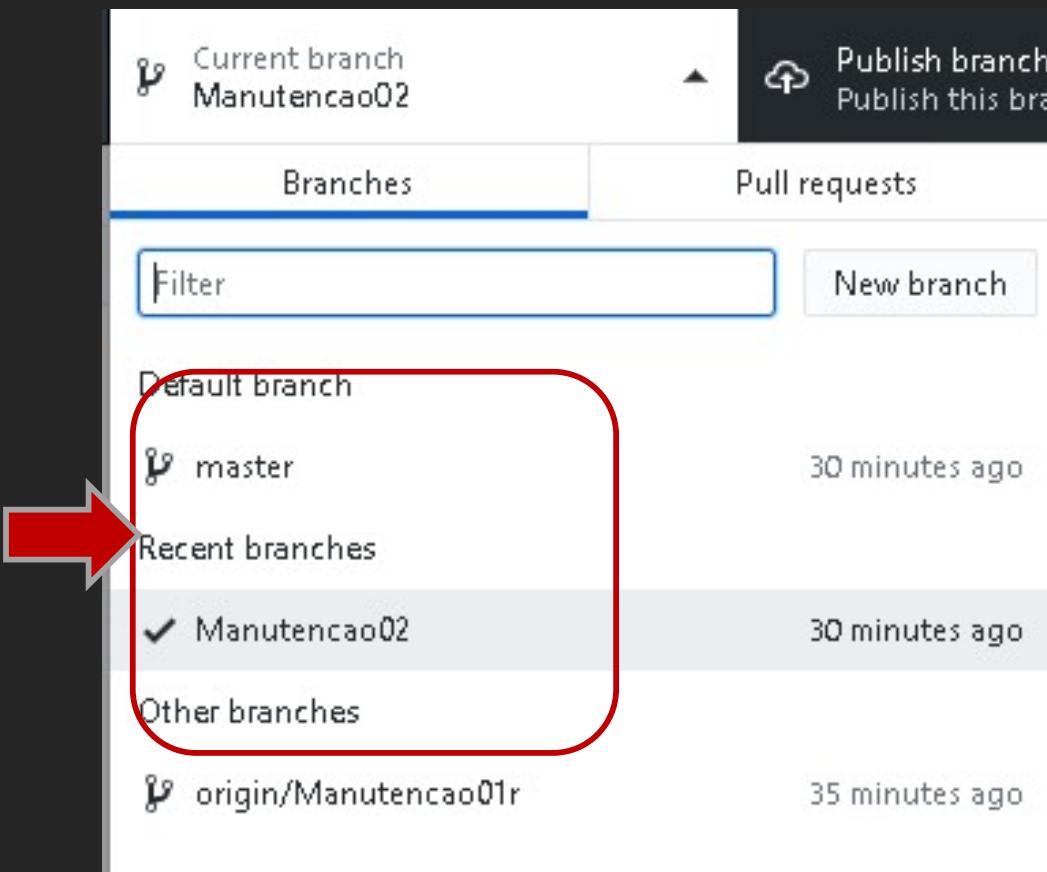
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Pode ainda criar branch...



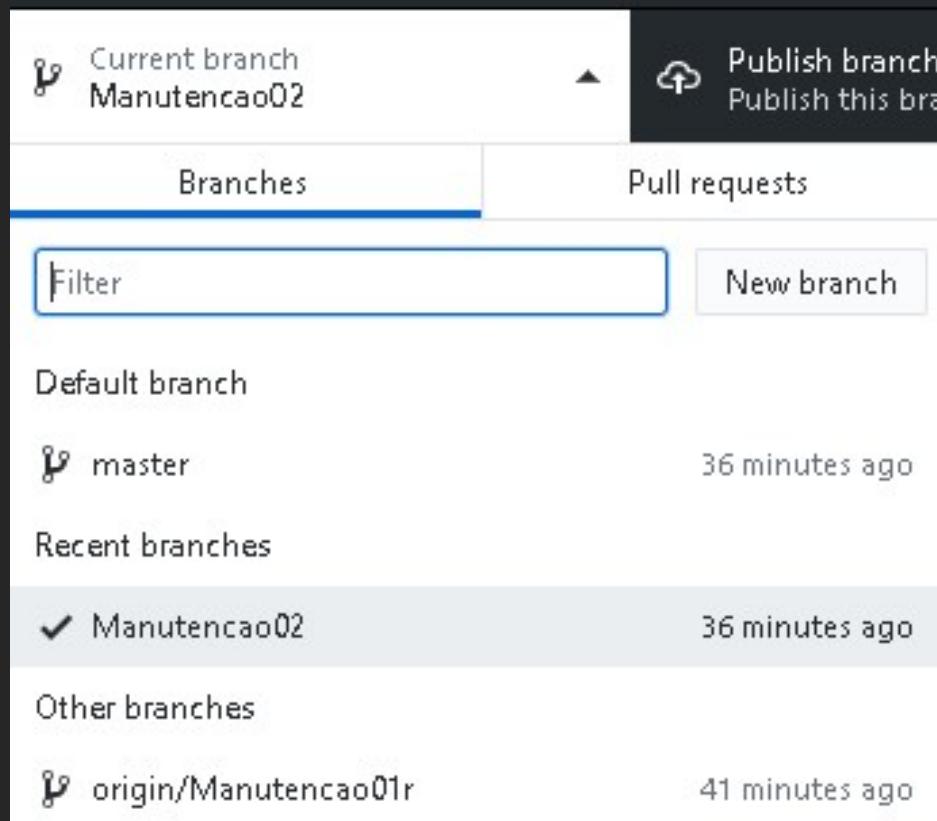
## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

E essa Branch quando criada localmente, pode ser visualizada no GitHub client.



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

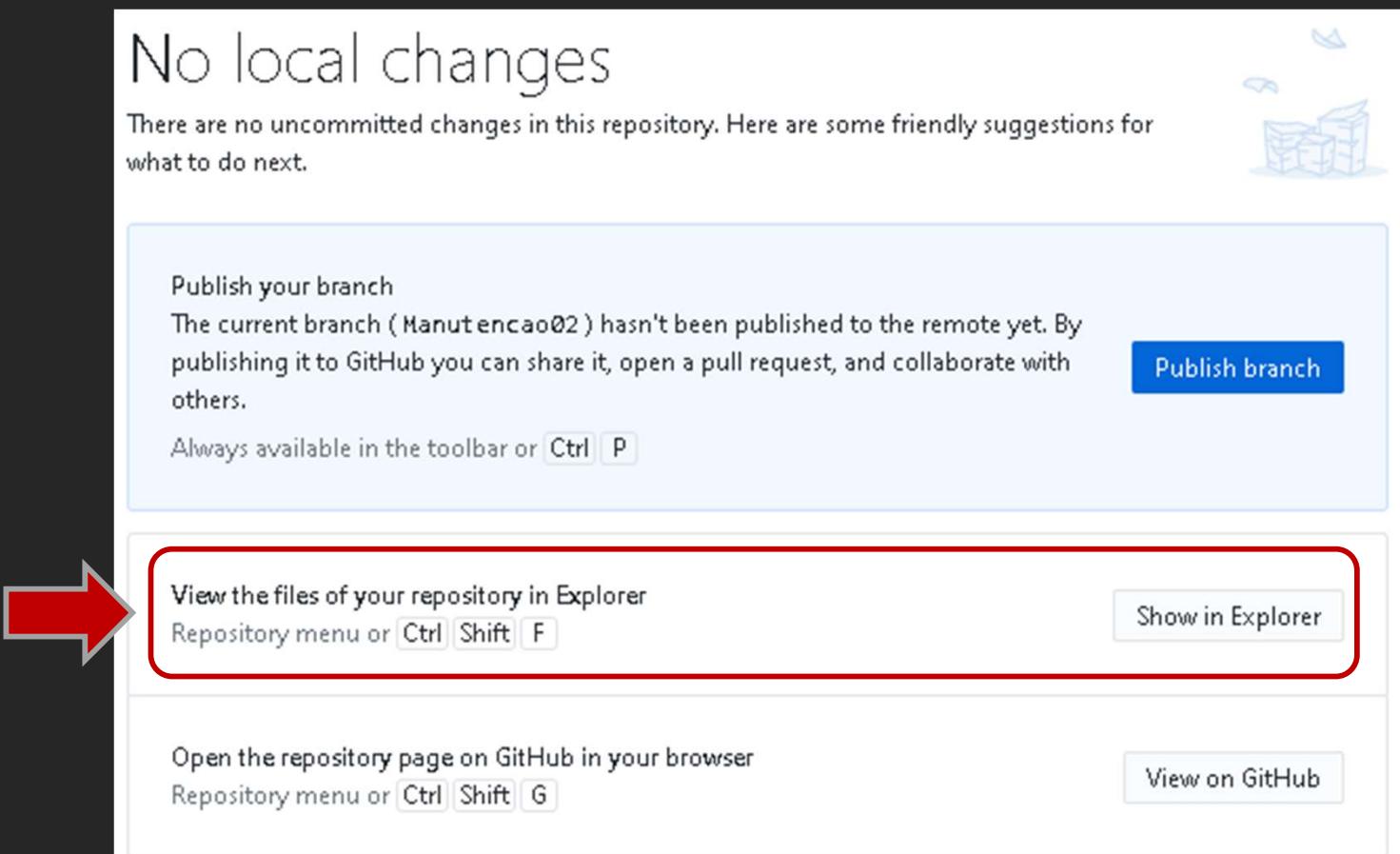
Para editar arquivos na Branch, faça um Checkout!



Selecione a branch no menu de navegação da ferramenta!

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Após criada a Branch, a navegação de acesso aos fontes é via menu da aplicação GUI.



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Assim que alterado o arquivo ou criado um arquivo novo na pasta do GIT local, o GitHub Client vai mostrar as diferenças entre a Master e a Branch.

The screenshot shows the GitHub Desktop application interface. At the top, there's a menu bar with File, Edit, View, Repository, Branch, and Help. Below the menu, there are three main sections: 'Current repository' (TesteGITgui), 'Current branch' (Manutencao02), and 'Publish branch' (Publish this branch to GitHub). The central area is titled 'Changes 1'. It shows a list of changes: '1 changed file' and 'README.md'. The 'History' tab is also visible. On the right, a detailed view of the README.md file's changes is shown in a diff editor. The changes are:

Line	Line	Change
1	1	@@ -1,2 +1,2 @@
2	2	# TesteGITgui
2	2	-# Texto NOVO
2	2	+# Texto NOVO MESMO!!!

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Se você desejar publicar a Branch criada no seu PC no GitHub.com, primeiro, faça um Commit no Git local



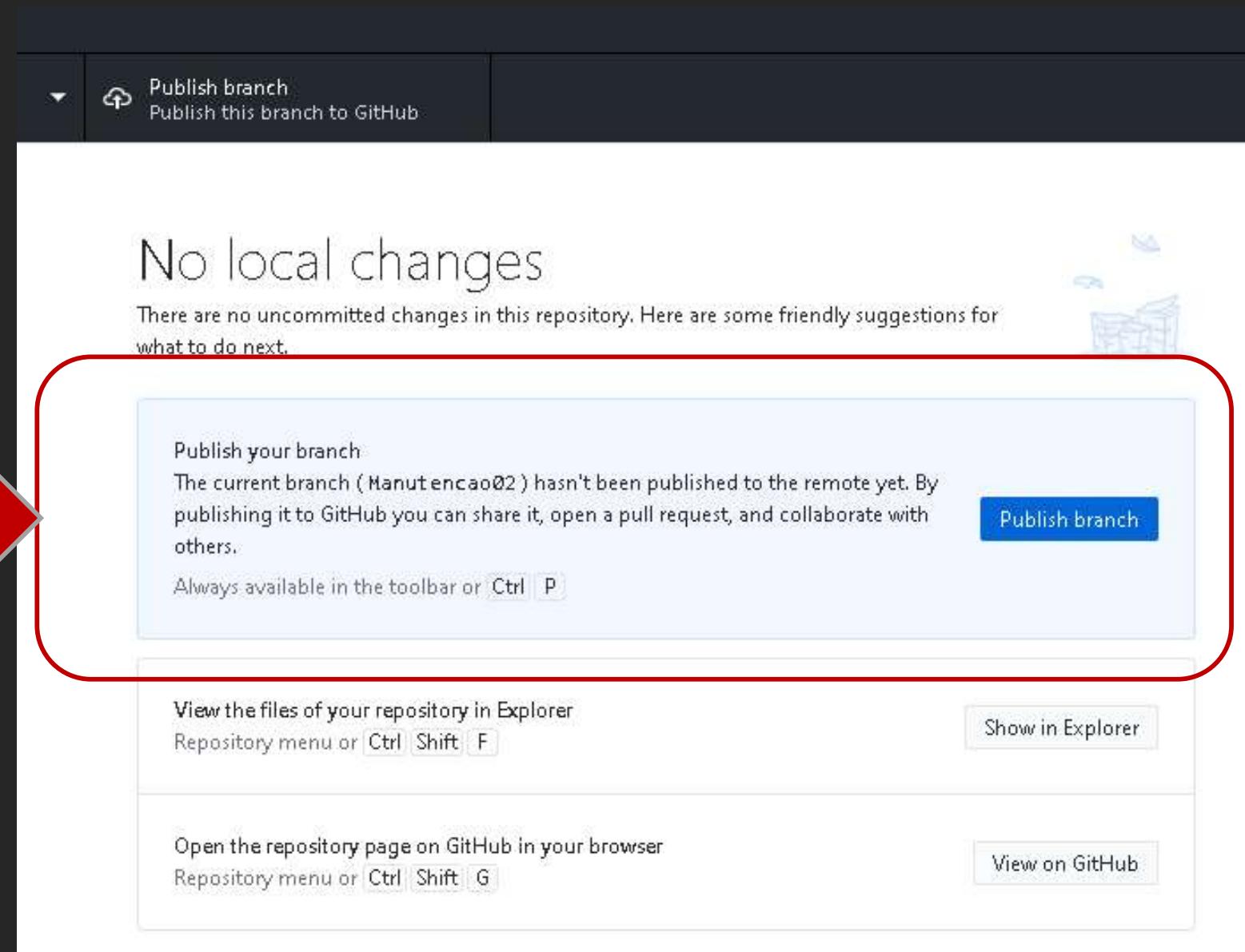
The screenshot shows a Git commit interface. At the top, it displays the current repository as 'TesteGITgui' and the current branch as 'Manutencao02'. A button for publishing the branch to GitHub is also visible. Below this, a table shows the changes made to 'README.md':

Changes	1	History
<input checked="" type="checkbox"/> README.md	1	1 @@ -1,2 +1,2 @@ # TesteGITgui -# Texto NOVO @@ +# Texto NOVO MESMO!!! @@

At the bottom, there is a section for updating the file and a 'Commit to Manutencao02' button.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Faça o PUSH no  
GITHUB remoto  
(WEB).



# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

As edições são refletidas do GitGui para o GitHub!

Crie a Pull Request remota via GITHUB Client para atualizar a Master remota.

The screenshot shows the GitHub desktop application interface. At the top, it displays a status message: "Fetch origin" with "Last fetched 2 minutes ago". Below this, the main area shows the message "No local changes". A descriptive text follows: "There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next." Three options are listed in boxes: 1) "Create a Pull Request from your current branch" (with a note that the current branch is already published to GitHub), accompanied by a "Create Pull Request" button. 2) "View the files of your repository in Explorer" (with a keyboard shortcut "Repository menu or Ctrl Shift F"), accompanied by a "Show in Explorer" button. 3) "Open the repository page on GitHub in your browser" (with a keyboard shortcut "Repository menu or Ctrl Shift G"), accompanied by a "View on GitHub" button. The background of the application window features a faint illustration of a city skyline.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Ao solicitar a Pull Request, o Git Hub é acionado. Você fará Login e confirmará o Merge/Pull Request!

The screenshot shows a GitHub pull request for a file named 'Update README.md' from branch '#2'. The pull request is from user 'RenatoJardimParducci' into the 'master' branch. It has one commit and one file changed. A comment from 'RenatoJardim Parducci' states: 'No description provided.' Below the commit, there is a green checkmark icon next to the file name 'Update README.md' and the commit hash '77a8b17'. A note at the bottom says: 'Add more commits by pushing to the Manutencao02 branch on RenatoJardimParducci/TesteGITgui.' A green box highlights two status indicators: 'Continuous integration has not been set up' (with a note about GitHub Actions) and 'This branch has no conflicts with the base branch' (with a note about automatic merging). At the bottom, there is a green button labeled 'Merge pull request' and a note: 'You can also open this in GitHub Desktop or view command line instructions.' A red arrow points from the right towards the 'Merge pull request' button.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

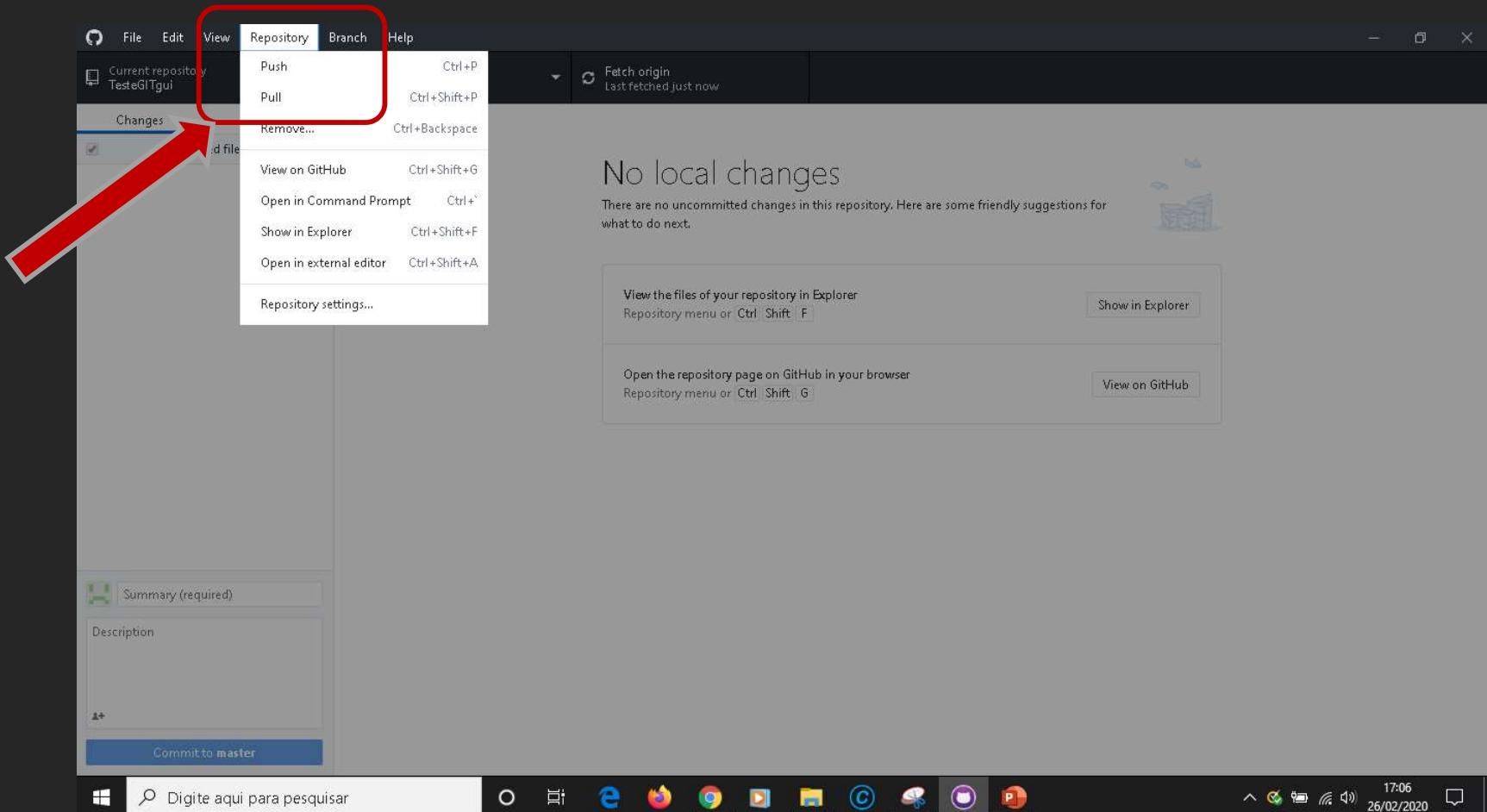
Com esses procedimentos, você consegue fazer Check out e Check in (Push) , mantendo sincronia entre o ambiente remoto dos fontes e o seu ambiente local de trabalho.

Após os Pushes, confira sempre se o GitHub.com foi atualizado corretamente.

The screenshot shows a GitHub repository interface. At the top, there are statistics: 5 commits, 3 branches, 0 packages, 0 releases, and 1 contributor. Below this, a navigation bar includes 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A list of recent activity shows a merge pull request from RenatoJardimParducci/Merge pull request #2 from RenatoJardimParducci/Manutencao02, which was merged 1 minute ago. It also shows an update to the README.md file 9 minutes ago. The README.md file content is displayed below, showing the text 'TesteGITgui' and 'Texto NOVO MESMMO!!!'.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

**IMPORTANTE:** o fato de você ter atualizado a Branch e Master no GITHUB não implica que a Master Local tenha sido atualizada.  
Para atualizar a Master Local, faça um Pull do Repositório.



ASSISTA OS VÍDEOS NO CANAL DO PROFESSOR



[https://youtu.be/ps\\_ZTFrasAg](https://youtu.be/ps_ZTFrasAg)

A screenshot showing a browser window with a GitHub repository page and a video player window side-by-side. The GitHub page displays a repository named 'RenatoJardimParducci / ExemploGitDesktop'. It shows 1 commit, 1 branch, 0 packages, and 0 releases. The README.md file contains the text 'ExemploGitDesktop'. To the right of the browser is a video player window showing a man with dark hair and a blue shirt, speaking. A small text at the bottom right of the video player says 'Double Click on image to play movie'.



## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Outra forma de usar o GIT é na linha de prompt do sistema operacional (opção que muitas vezes é a favorita de alguns desenvolvedores).

Para isso, instale o GIT CMD na sua máquina e ao rodá-lo, a tela de prompt do sistema operacional aparecerá e você poderá executar comandos nesse prompt.

A lista de comandos que podem ser usados via prompt estão relacionados a seguir.



# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Lista de comandos do GIT CMD (considerando que o repositório já existe no GIT HUB):

## Setar usuário

```
git config --global user.name "Seu nome"
```

## Criando um novo branch

```
git branch bug-123
```

## Setar email

```
git config --global user.email Seu e-mail (sem aspas)
```

## Trocando para um branch existente

```
git checkout bug-123
```

## Clonar um repositório remoto já existente

```
git clone Seu URL de GitHub
```

## Apagando um branch

```
git branch -d bug-123
```

## Verificar estado dos arquivos/diretórios

```
git status
```

## Listar branches

## Listar branches

```
git branch
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Lista de comandos do GIT CMD (considerando que o repositório já existe no GIT HUB):

Atualizar os arquivos no branch atual

```
git pull
```

Comitar arquivo/diretório

Comitar um arquivo

```
git commit meu_arquivo.txt
```

Adicionar arquivo/diretório (staged area)

Adicionar um arquivo em específico

```
git add meu_arquivo.txt
```

Comitar vários arquivos

```
git commit meu_arquivo.txt meu_outro_arquivo.txt
```

Adicionar um diretório em específico

```
git add meu_diretorio
```

Comitar informando mensagem

```
git commit meu_arquivo.txt -m "minha mensagem de commit"
```

Adicionar todos os arquivos/diretórios

```
git add .
```

Exibir histórico modificação de um arquivo

```
git log --diff-filter=M -- <caminho_do_arquivo>
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Lista de comandos do GIT CMD (considerando que o repositório já existe no GIT HUB):

## Remover arquivo/diretório

### Remover arquivo

```
git rm meu_arquivo.txt
```

### Desfazendo alteração local (staging area)

Este comando deve ser utilizado quando o arquivo já foi adicionado na **staged area**.

```
git reset HEAD meu_arquivo.txt
```

### Remover diretório

```
git rm -r diretorio
```

Se o resultado abaixo for exibido, o comando reset *não* alterou o diretório de trabalho.

```
Unstaged changes after reset:  
M     meu_arquivo.txt
```

## Visualizar histórico

### Exibir histórico

```
git log
```

### Exibir histórico de um arquivo específico

```
git log -- <caminho_do_arquivo>
```

## Enviar arquivos/diretórios para o repositório remoto

O primeiro **push** de um repositório deve conter o nome do repositório remoto e o branch.

```
git push -u origin master
```

Coloque o nome da Branch  
que você atualizou para ela  
ser levada ao GITHUB !!

Os demais **pushes** não precisam dessa informação

```
git push
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Lista de comandos do GIT CMD (considerando que o repositório já existe no GIT HUB):

Desconectar do GIT remoto:

```
git config --global --unset user.name  
git config --global --unset user.email
```

Ou:

```
git config --global --unset-all
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

Lista de comandos do GIT CMD (considerando que o repositório já existe no GIT HUB):

Se você desejar desfazer uma versão/atualização específica:

```
git revert 872fa7e
```

ID da versão a ser desfeita

Se você quer voltar para um ponto da história, esquecendo o que aconteceu depois:

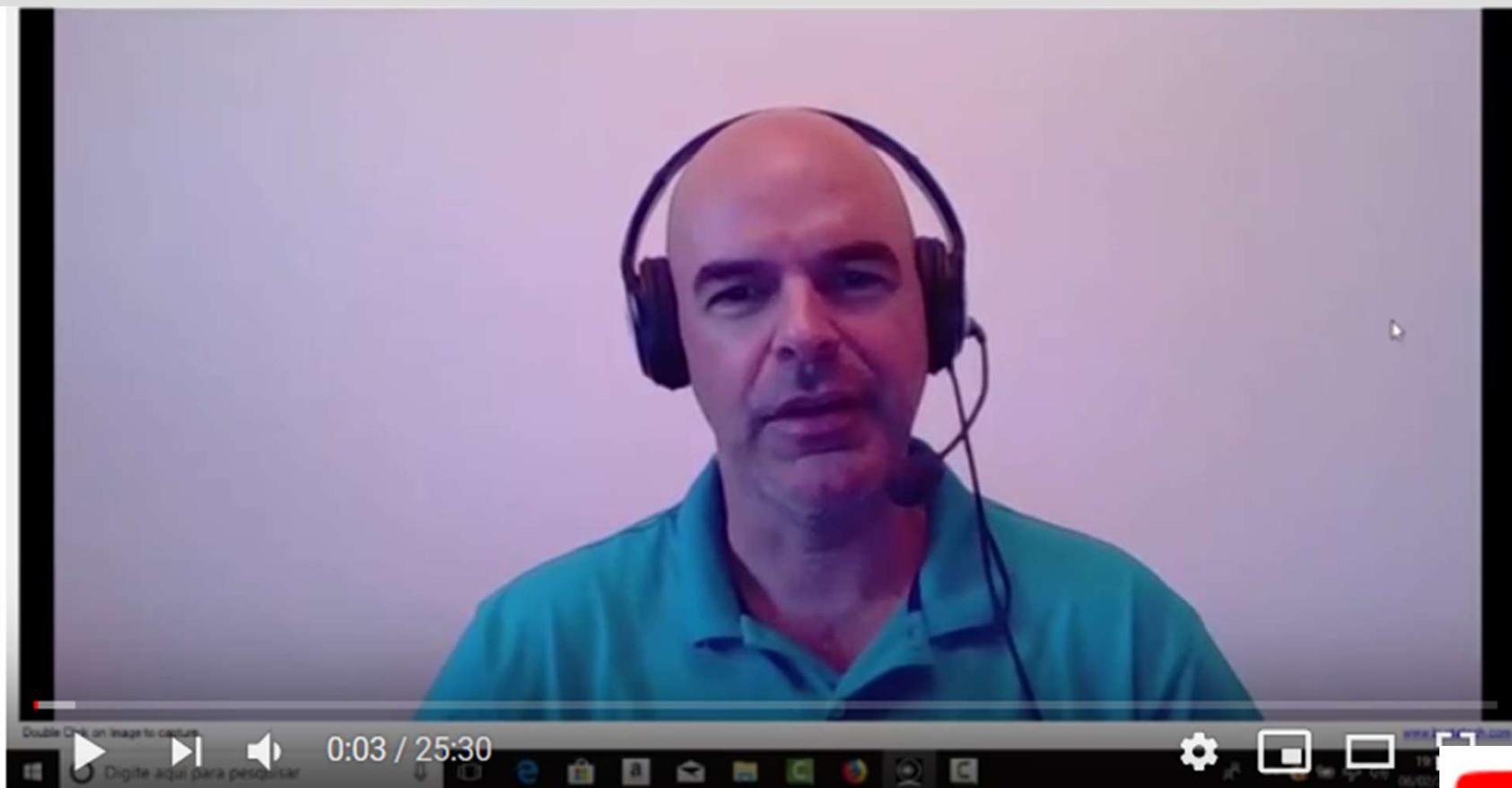
```
git reset –hard a1e8fb5
```

ID da versão a ser  
reestabelecida/recuperada

ASSISTA OS VÍDEOS NO CANAL DO PROFESSOR



[https://youtu.be/aSpdQ\\_82S9k](https://youtu.be/aSpdQ_82S9k)



Conteúdo didático complementar - GIT-CMD

*APROVEITE PARA CONTROLAR FONTES DE PROGRAMAS E DOCUMENTOS DE PROJETOS COM GIT/GIT HUB, DAQUI POR DIANTE!*



ASSISTA OS VÍDEOS NO CANAL DO PROFESSOR

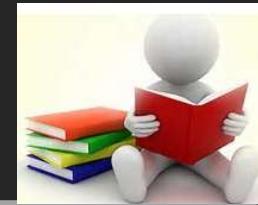


<https://youtu.be/wB6aLIwcb00>



GIT - Simulação completa com HUB, GUI, Desktop e CMD

## ESTUDO DE CASO SIMULADO



A GD (Gerência de Desenvolvimento) da empresa de Dilan começou a experimentar o GIT. Algumas equipes preferem operar o GITHIUB diretamente, outras via cliente GitHubDesktop, outras preferem o GitGUI e outras preferem o GitCMD.

Com o uso das ferramentas, surgiu um problema: algumas vezes os desenvolvedores alteravam diretamente a cópia Master, outras vezes, sentiam falta de poder abrir várias Branchs de desenvolvimento à partir de um ambiente/Brench de manutenção geral, de forma a possibilitar testes integrados ao final dos desenvolvimentos de todos programadores para atender a uma Release de versão.

Você descobriu uma forma de trabalhar com o GIT para controlar melhor o fluxo de trabalho, usando o GIT FLOW e vai realizar uma atividade prática para trabalhar com seus colegas e ensiná-los a tecnologia aplicada ao gerenciamento de fontes e documentos de sistemas em desenvolvimento.

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

Quando trabalhamos com grandes projetos ou quando temos vários programadores desenvolvendo partes de uma mesma solução que precisa ser integrada ao final, o ideal é trabalhar com um modelo de Branches em cascata (Branch de Branch), , separando os fontes estáveis dos instáveis e daqueles que ainda precisam de uma última bateria de testes.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

O GitFlow é um conjunto de programas embarcados no GIT CMD e permite o controle automático de situações que envolvem múltiplos níveis de Branch.

Ele evita esquecimentos e atualizações erradas. Impede que manutenções que ainda não estão efetivamente validadas sejam movidas para o diretório de fontes que são parte do produto de software estável, o qual é liberado para usuários.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

Vamos trabalhar com as seguintes BRANCHES que são apropriadas para grandes projetos:

- ⇒ ReleaseManufacture (nossa Master onde estão os fontes prontos para liberação para uso);
- ⇒ ReleaseCandidate (contém os fontes liberados pelos desenvolvedores para testes finais). Essa BRANCH acumula as mudanças promovidas por todos programadores no software;
- => Development-NOME DA TAREFA DE DESENVOLVIMENTO (contém os fontes que estão ainda em programação e testes pelos desenvolvedores). Essas BRANCHES devem ser excluídas quando a tarefa de produção do desenvolvedor acabar.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Execute o Git CMD

-Vá até uma pasta onde criaremos uma sub pasta com o nome gitflowExemplo (use os comandos do DOS – MD, CD, DIR para criar e conferir a criação)

ALTERNATIVAMENTE, VOCÊ PODE CLONAR UM REPOSITÓRIO GITHUB OU ADICIONAR DEPOIS ESSE LINK COM O HUB NA PASTA CRIADA.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Execute o comando git flow init para colocar o novo repositório dentro do controle GIT

```
C:\Users\renat\gitflowExemplo>git flow init
Initialized empty Git repository in C:/Users/renat/gitflowExemplo/.git/
No branches exist yet. Base branches must be created now.
```

-Aguarde a criação da pasta e depois, informe o nome da BRANCH que conterá os fontes na versão pronta para gerar a Release e fazer Deploy para liberar o uso (o nome default é Master e vamos renomear para ReleaseManufacture)

```
Branch name for production releases: [master] ReleaseManufacture
```

AGUARDE O PROMPT PEDINDO A DEFINIÇÃO DA BRANCH DE  
DESENVOLVIMENTO

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Nomeie em seguida a BRANCH onde estarão os fontes prontos para testes finais pré-liberação  
(o nome default é Development e vamos renomear para ReleaseCandidate)

```
Branch name for "next release" development: [develop] ReleaseCandidate
```

AGUARDE O PROMPT PEDINDO A DEFINIÇÃO DOS LABELS DE PREFIXO  
DAS BRANCHES DE FEATURE, BUGFIX, HOTFIX, RELEASE, SUPPORT.

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

-Deixe o padrão feature/ como prefixo de BRANCHES de manutenção (já sugerido pelo Git Flow) ou mude, se desejar e aceite o Bugfix (dê ENTER) e não vamos acrescentar TAGs

```
C:\Users\renat\gitflowExemplo>git flow init
Initialized empty Git repository in C:/Users/renat/gitflowExemplo/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master] ReleaseManufacture
Branch name for "next release" development: [develop] ReleaseCandidate

How to name your supporting branch prefixes?
Feature branches? [feature/] DevelopmentFeatureTesteFlow
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/renat/gitflowExemplo/.git/hooks]
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Ao terminar, o GitFlow vai posicionar você, automaticamente, no diretório de manutenção/feature

```
C:\Users\renat\gitflowExemplo>
```

-Veja as Branches existentes com o comando git branch

```
C:\Users\renat\gitflowExemplo>git branch
* ReleaseCandidate
  ReleaseManufacture
```

Perceba que dentro da sua pasta de tarefa chamada gitflowExemplo, existe a Branch ReleaseCandidate (Master) e a ReleaseManufacture (fontes liberados par os testes finais), as quais você pode usar.

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Crie agora uma feature que corresponde àquilo que você como desenvolvedor precisa realizar para cumprir uma tarefa de programação ou documentação de software que lhe foi designada

```
C:\Users\renat\gitflowExemplo>git flow feature start DocumentacaoProjeto
```

```
Summary of actions:
```

```
- A new branch 'DevelopmentFeatureTesteFlowDocumentacaoProjeto' was created, based on 'ReleaseCandidate'  
- You are now on branch 'DevelopmentFeatureTesteFlowDocumentacaoProjeto'
```

```
Now, start committing on your feature. When done, use:
```

```
git flow feature finish DocumentacaoProjeto
```

-Confira a criação da Branch e veja que o GitFlow já posicionou você na Branch de programação/documentação.

```
C:\Users\renat\gitflowExemplo>git branch  
* DevelopmentFeatureTesteFlowDocumentacaoProjeto  
  ReleaseCandidate  
  ReleaseManufacture
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

-Agora, você pode usar editores de linguagem ou documentos, navegar via Windows Explorer, por exemplo e copiar, editar, criar arquivos à vontade. O Git vai mapear tudo o que você fizer no diretório do projeto, vinculando com a Branch de tarefa

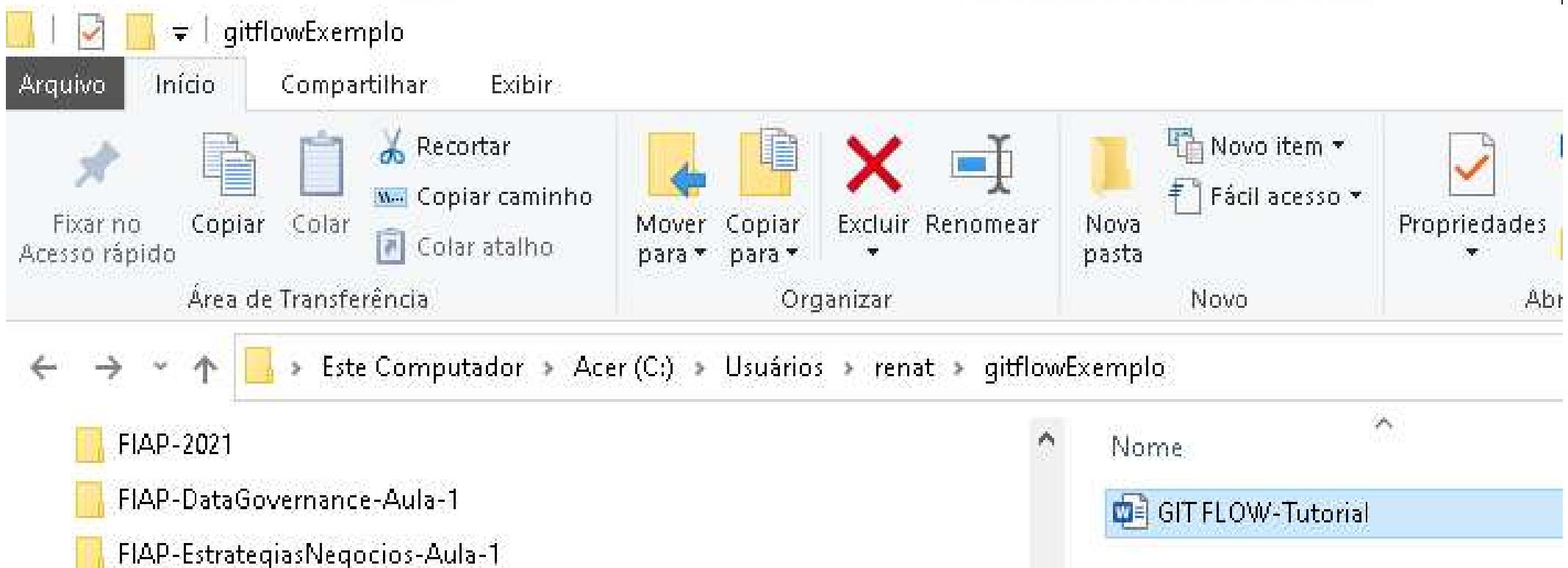
C:\Users\renat\gitflowExemplo>



# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

-Experimente copiar um arquivo qualquer para dentro da pasta e daí para frente, podemos aplicar os comandos do Git Command já estudados (status, add, commit, push, pull, ...)



# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

-Git status identificará as mudanças em arquivos

```
C:\Users\renat\gitflowExemplo>git status
On branch DevelopmentFeatureTesteFlowDocumentacaoProjeto
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    GIT FLOW-Tutorial.docx

nothing added to commit but untracked files present (use "git add" to track)
```

-Adicione o arquivo para coloca-lo na lista de arquivos prontos para commit, com o Git add

```
C:\Users\renat\gitflowExemplo>git add .
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Confira se o arquivo foi adicionado com Git status

```
C:\Users\renat\gitflowExemplo>git status
On branch DevelopmentFeatureTesteFlowDocumentacaoProjeto
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   GIT FLOW-Tutorial.docx
```

-Agora, faça o commit para que o GitFlow publique o seu novo arquivo/versão nova, na branch de testes finais

```
C:\Users\renat\gitflowExemplo>git commit -m "Adicao do documento de kickoff do projeto"
[DevelopmentFeatureTesteFlowDocumentacaoProjeto 4920569] Adicao do documento de kickoff do projeto
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 GIT FLOW-Tutorial.docx
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

- Informe agora que você finalizou o desenvolvimento da sua tarefa/feature para que o Gitflow leve as atualizações sozinho para a Branch “superior” – a Branch de testes finais chamada ReleaseCandidate. Para isso, execute o comando Git flow feature finish nome da feature/branch terminada

```
C:\Users\renat\gitflowExemplo>git flow feature finish DocumentacaoProjeto
Switched to branch 'ReleaseCandidate'
Updating 347b852..4920569
Fast-forward
  GIT FLOW-Tutorial.docx | Bin 0 -> 73308 bytes
    1 file changed, 0 insertions(+), 0 deletions(-)
     create mode 100644 GIT FLOW-Tutorial.docx
Deleted branch DevelopmentFeatureTesteFlowDocumentacaoProjeto (was 4920569).

Summary of actions:
- The feature branch 'DevelopmentFeatureTesteFlowDocumentacaoProjeto' was merged into 'ReleaseCandidate'
- Feature branch 'DevelopmentFeatureTesteFlowDocumentacaoProjeto' has been locally deleted
- You are now on branch 'ReleaseCandidate'
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-o GitFlow já deleta a sua Branch de manutenção/tarefa e atualiza a Branch superior. Ele também posiciona você na Branch superior. Confira as mudanças na Branch superior com Git branch e Git status

```
C:\Users\renat\gitflowExemplo>git branch
* ReleaseCandidate
  ReleaseManufacture
```

```
C:\Users\renat\gitflowExemplo>git status
On branch ReleaseCandidate
nothing to commit, working tree clean
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

Perceba que o arquivo que você alterou/criou/excluiu, já foi modificado e “comitado” na Branch superior. Confira com o comando Dir

```
C:\Users\renat\gitflowExemplo>dir
O volume na unidade C é Acer
O Número de Série do Volume é 2671-A4C5

Pasta de C:\Users\renat\gitflowExemplo

07/03/2021  10:36    <DIR>          .
07/03/2021  10:36    <DIR>          ..
07/03/2021  10:36           73.308 GIT FLOW-Tutorial.docx
                           1 arquivo(s)      73.308 bytes
                           2 pasta(s)   738.099.294.208 bytes disponíveis
```

-Se você executar git flow feature start de novo, o GitFlow vai criar uma nova Branch de manutenção com as últimas versões da ReleaseCandidate

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Quando as manutenções ligadas a uma Release acabarem, você deve disparar a geração da Release com o comando Git flow release start *número da release*

```
C:\Users\renat\gitflowExemplo>git flow release start R01.01
Switched to a new branch 'release/R01.01'

Summary of actions:
- A new branch 'release/R01.01' was created, based on 'ReleaseCandidate'
- You are now on branch 'release/R01.01'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish 'R01.01'
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Nesse momento o GitFlow criou uma Branch de release para que o software receba a carga de testes finais, pré-liberação. Confira com Git branch.

```
C:\Users\renat\gitflowExemplo>git branch
  ReleaseCandidate
  ReleaseManufacture
* release/R01.01
```

Quando você acessar os arquivos da pasta de projeto, agora estará operando sobre a branch da release

```
C:\Users\renat\gitflowExemplo>dir
O volume na unidade C é Acer
O Número de Série do Volume é 2671-A4C5

Pasta de C:\Users\renat\gitflowExemplo

07/03/2021 10:36    <DIR>      .
07/03/2021 10:36    <DIR>      ..
07/03/2021 10:36            73.308 GIT FLOW-Tutorial.docx
                           1 arquivo(s)      73.308 bytes
                           2 pasta(s)   738.100.359.168 bytes disponíveis
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Quando acabar os testes, feche a Release de forma parecida com o que fez com a Feature: Git flow release finish nome da release a fechar -m "mensagem que descreve a release"

```
C:\Users\renat\gitflowExemplo>git flow release finish R01.01 -m "100% testes finalizados da 1a Release"
Switched to branch 'ReleaseManufacture'
Switched to branch 'ReleaseCandidate'
Already up to date!
Merge made by the 'recursive' strategy.
Deleted branch release/R01.01 (was 4920569).

Summary of actions:
- Release branch 'release/R01.01' has been merged into 'ReleaseManufacture'
- The release was tagged 'R01.01'
- Release tag 'R01.01' has been back-merged into 'ReleaseCandidate'
- Release branch 'release/R01.01' has been locally deleted
- You are now on branch 'ReleaseCandidate'
```

-Agora, a Branch da Release é destruída e você vai para a ReleaseCandidate, automaticamente.  
Confira com o comando Git branch

```
C:\Users\renat\gitflowExemplo>
C:\Users\renat\gitflowExemplo>git branch
* ReleaseCandidate
  ReleaseManufacture
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-OBS: SE VOCÊ ESQUECER DE COLOCAR O -m “MENSAGEM”, O GIT ABRIRÁ UM EDITOR PARA VOCÊ COLOCAR OS COMENTÁRIOS. ADICIONE O QUE DESEJAR NO TEXTO E DEPOIS DIGITE “ESC” E EM SEGUIDA “:wq!” E ENTER DE NOVO.

-Execute Git tag para verificar que o Git já vinculou a identificação da Release com o conteúdo que se encontra na ReleaseCandidate.

```
C:\Users\renat\gitflowExemplo>git tag  
R01.01
```

```
C:\Users\renat\gitflowExemplo>git branch  
* ReleaseCandidate  
  ReleaseManufacture
```

## GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

### GIT FLOW APLICADO

-Quando você terminou a Branch de release, o GitFlow TAMBÉM ATUALIZOU A MASTER (ReleaseManufacture). Confira, mudando para a Branch ReleaseManufacture e listando os arquivos com DIR. Você pode também usar o Explorer do Windows para conferir os arquivos publicados na cópia mestre.

```
C:\Users\renat\gitflowExemplo>git checkout ReleaseManufacture
Switched to branch 'ReleaseManufacture'

C:\Users\renat\gitflowExemplo>dir
O volume na unidade C é Acer
O Número de Série do Volume é 2671-A4C5

Pasta de C:\Users\renat\gitflowExemplo

07/03/2021  11:04    <DIR>          .
07/03/2021  11:04    <DIR>          ..
07/03/2021  11:04                73.308 GIT FLOW-Tutorial.docx
                           1 arquivo(s)   73.308 bytes
                           2 pasta(s)   738.099.748.864 bytes disponíveis
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

-Se você precisar fazer um **HOTFIX** em um fonte que já está em produção (correção rápida de um BUG), você pode criar uma Branch diretamente à partir da Master e publicar diretamente na Master quando terminar.

```
C:\Users\renat\gitflowExemplo>git flow hotfix start CorrecaoKickoff
Switched to a new branch 'hotfix/CorrecaoKickoff'
```

Summary of actions:

- A new branch 'hotfix/CorrecaoKickoff' was created, based on 'ReleaseManufacture'
- You are now on branch 'hotfix/CorrecaoKickoff'

Follow-up actions:

- Start committing your hot fixes
- Bump the version number now!
- When done, run:

```
git flow hotfix finish 'CorrecaoKickoff'
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

-O Git vai mover você para o diretório de correção. Confira com Git branch.

```
C:\Users\renat\gitflowExemplo>git branch
  ReleaseCandidate
  ReleaseManufacture
* hotfix/CorrecaoKickoff
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

-Modifique o que precisa (execute `git add` e `commit` para confirmar as mudanças)

```
C:\Users\renat\gitflowExemplo>git status
On branch hotfix/CorrecaoKickoff
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   GIT FLOW-Tutorial.docx

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

C:\Users\renat\gitflowExemplo>git diff
diff --git a/GIT FLOW-Tutorial.docx b/GIT FLOW-Tutorial.docx
index d67755a..064f85b 100644
--- a/GIT FLOW-Tutorial.docx
+++ b/GIT FLOW-Tutorial.docx
@@ -1,4 +1,4 @@
-GIT FLOW
+GIT FLOW - TUTORIAL EXTRA
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

```
C:\Users\renat\gitflowExemplo>git add .

C:\Users\renat\gitflowExemplo>git status
On branch hotfix/CorrecaoKickoff
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   GIT FLOW-Tutorial.docx

C:\Users\renat\gitflowExemplo>git commit -m "Adicao de detalhes no documento"
[hotfix/CorrecaoKickoff b82793a] Adicao de detalhes no documento
  1 file changed, 1 insertion(+)
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

- Ao terminar, finalize o Hot fix com Git flow hotfix finish -m "mensagem sobre a manutenção"

```
C:\Users\renat\gitflowExemplo>git flow hotfix finish -m "Texto de kickoff corrigido"
Switched to branch 'ReleaseManufacture'
Merge made by the 'recursive' strategy.
  GIT FLOW-Tutorial.docx | Bin 73308 -> 73133 bytes
    1 file changed, 0 insertions(+), 0 deletions(-)
Switched to branch 'ReleaseCandidate'
Merge made by the 'recursive' strategy.
  GIT FLOW-Tutorial.docx | Bin 73308 -> 73133 bytes
    1 file changed, 0 insertions(+), 0 deletions(-)
Deleted branch hotfix/CorrecaoKickoff (was b82793a).
```

Summary of actions:

- Hotfix branch 'hotfix/CorrecaoKickoff' has been merged into 'ReleaseManufacture'
- The hotfix was tagged 'CorrecaoKickoff'
- Hotfix tag 'CorrecaoKickoff' has been back-merged into 'ReleaseCandidate'
- Hotfix branch 'hotfix/CorrecaoKickoff' has been locally deleted
- You are now on branch 'ReleaseCandidate'

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

Veja de a Branch de hotfix foi excluída automaticamente.

-**Após finalizar o Hot fix, o Git flow atualiza automaticamente a cópia Master (nossa ReleaseManufacture) e a cópia de teste finais (ReleaseCandidate), da mesma forma que acontece quando você finaliza um desenvolvimento normal, usando releases tradicionais (não hot fix). Você pode checar isso, navegando de uma branch para outra e observando o horário de atualização do arquivo corrigido e/ou observando o seu conteúdo.**

```
C:\Users\renat\gitflowExemplo>git branch
* ReleaseCandidate
  ReleaseManufacture

C:\Users\renat\gitflowExemplo>
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

```
C:\Users\renat\gitflowExemplo>dir
O volume na unidade C é Acer
O Número de Série do Volume é 2671-A4C5

Pasta de C:\Users\renat\gitflowExemplo

07/03/2021  11:38    <DIR>          .
07/03/2021  11:38    <DIR>          ..
07/03/2021  11:38            73.133 GIT FLOW-Tutorial.docx
                           1 arquivo(s)   73.133 bytes
                           2 pasta(s)   738.106.187.776 bytes disponíveis

C:\Users\renat\gitflowExemplo>git checkout ReleaseManufacture
Switched to branch 'ReleaseManufacture'

C:\Users\renat\gitflowExemplo>dir
O volume na unidade C é Acer
O Número de Série do Volume é 2671-A4C5

Pasta de C:\Users\renat\gitflowExemplo

07/03/2021  11:38    <DIR>          .
07/03/2021  11:38    <DIR>          ..
07/03/2021  11:38            73.133 GIT FLOW-Tutorial.docx
                           1 arquivo(s)   73.133 bytes
                           2 pasta(s)   738.106.187.776 bytes disponíveis
```

# GERENCIAMENTO DA PRODUÇÃO E ATUALIZAÇÃO DE SOFTWARE

## GIT FLOW APLICADO

Agora você conhece ferramentas e melhores práticas mais populares e eficientes para gerenciar programas e documentos durante o desenvolvimento de um software



D Ú V I D A S

## Referência bibliográficas

### BIBLIOGRAFIA:

- **ISACA.** USA, COBIT 5 . 2014 - Disponível para acesso online gratuito em ISACA.org.
- **WEILL**, Peter. ROSS Jeane W. Governança de TI. Makron Books.
- **PRESSMAN**, Roger S.. Engenharia de software. - Uma abordagem profissional, 7<sup>a</sup> edição. São Paulo, AMGH.
- **HIRAMA, Kechi.** Engenharia de Software: qualidade e produtividade com tecnologia. Editora Elsevier, Rio de Janeiro.
- **BOEHM, Barry.** Software Engineering Economics. Prentice Hall, USA.

