Fundamentos de Desenvolvimento de Software 1

Prof. Pablo Vargas
Projeto Palomakoba

Tópicos Abordados

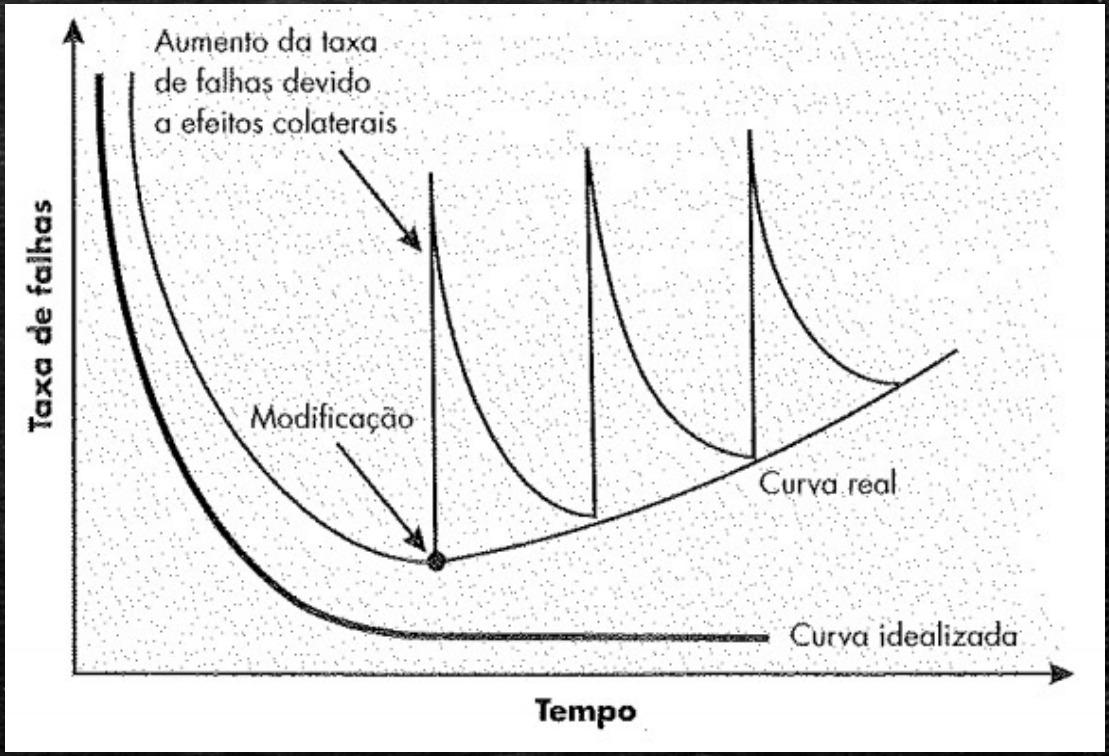
- Introdução à Engenharia de Software
- Desenvolvimento Ágil
- Gerenciamento de Projetos com Scrum
- Controle de Versão usando Git

Software e Engenharia de Software

- **Software** é tudo o que você xinga?
  - "É o **produto** que os profissionais de software **constroem e mantêm** ao longo do tempo." (Pressman)
  - Engenheiros de software, programadores, analistas, gerentes.
- O que é a **engenharia de software**?
  - "é uma **disciplina** da engenharia que se ocupa de **todos os aspectos da produção de software**." (Sommerville)

O que é Software?

- O Software não é **manufaturado.**
- O Software **deteriora-se.**
- Componentes de Software são “peças” reutilizáveis em outros programas.
- Parte **lógica** do computador.
- São programas, rotinas ou conjunto de instrução.



Mitos do Software

- “Já temos um manual repleto de padrões e procedimentos para a construção de software. Isso já é suficiente para o pessoal do desenvolvimento”.
- “Meu pessoal tem ferramentas de última geração, afinal de contas compramos os mais novos computadores”.
- “Se nós estamos atrasados nos prazos, podemos adicionar mais programadores e tirar o atraso”.
- “Uma declaração geral dos objetivos é suficiente para se começar a escrever programas, podemos preencher os detalhes mais tarde”.
- “Se terceirizar o projeto de software meus problemas acabam”.

Mitos do Software

- “Os requisitos de projeto modificam-se continuamente, mas as mudanças podem ser facilmente acomodadas, porque o software é flexível”.
- “Assim que escrevermos o programa e o colocarmos em funcionamento, nosso trabalho estará completo”.
- “Enquanto não tiver o programa funcionando, eu não terei realmente nenhuma maneira de avaliar sua qualidade”.
- “A única coisa a ser entregue em um projeto bem-sucedido é o programa funcionando”.
- “A engenharia de software vai me obrigar a criar documentação volumosa e desnecessária que certamente me atrasará”.

Processo

- O Processo de Software é um **conjunto de atividades, métodos, práticas e transformações ordenadas** com a intenção de atingir a **qualidade do software.**
- Sua meta fundamental é **entregar**, de maneira eficiente e previsível **um produto** de software **capaz de atender as necessidades** de negócio, definidas pela análise de requisitos, dos usuários.
- "Um processo define QUEM está fazendo O QUE, QUANDO e COMO para atingir esse objetivo." (Jacobson, Booch e Rumbaugh)
- **Estabilidade, controle e organização.**

Processo

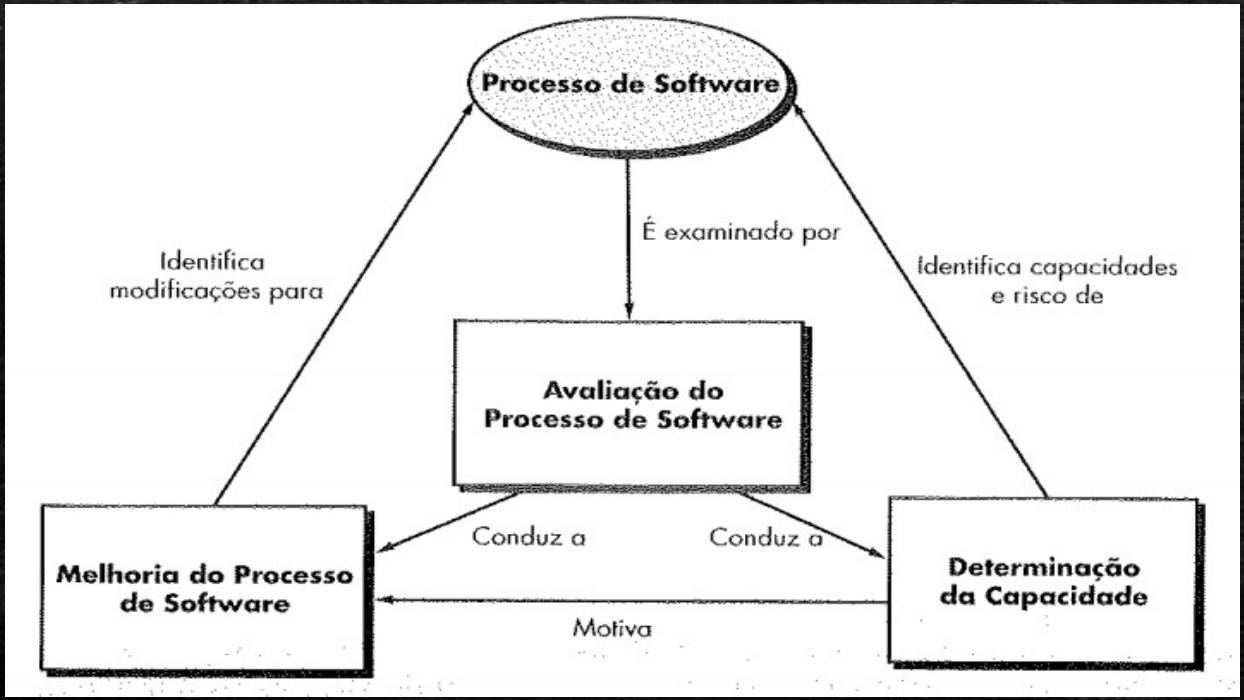
Processos, Métodos e Ferramentas
Ferramentas automatizadas ou semi-automatizadas para apoio ao desenvolvimento.
Orientações técnicas para a construção do software. Compreendem uma série de atividades, inclusive modelagem.
Ferramentas
Métodos
Processo
Foco na Qualidade
Agrega as diversas tecnologias para a construção de software. Organiza as tarefas, a aplicação dos métodos e o uso das ferramentas.
Base de tudo. Fator motivador para a Engenharia de Software.

Processo

- Atividades Fundamentais:
  - **Especificação, projeto, implementação, validação e evolução.**
- Software pode ser desenvolvido do inicio ou a partir de um software já existente.
- Foi uma das respostas técnicas adequadas para resolver a **crise do software.**

Avaliação de Processo

- Tenta entender **o estado atual** do processo de software.
- Conduz **a melhoria** no processo.



Produto x Processo

- Quando o **processo é fraco** o **produto** gerado por ele tende a ser de **baixa qualidade.**
- Quando o **processo é muito complexo** também pode acarretar em **esforço indevido.**
- Processo e produtos **são diretamente relacionados.**
- Quanto **menos linha de código** mais **flexível e melhor será o produto.**

Modelos de Processo de Software

- "...define um **conjunto** distinto de **atividades**, **ações**, **tarefas**, **marcos** e **produtos** de trabalho..." (Pressman)
- **Modelos** de software **devem ser adaptados** pelos engenheiros e gerentes a suas necessidades.
- **Ordem e consistência** devem ser o ponto fundamental do software a ser construído.

Modelos de Processo de Software

- Descrevem, basicamente, as principais etapas do desenvolvimento de software, desde a produção até a sua própria manutenção.
- Outras terminologia:
  - Modelos Prescritivos de Processo.
  - Paradigmas do Ciclo de Vida.
  - Paradigmas do Desenvolvimento de Software.
  - Modelagem do Ciclo de Vida.

Modelos de Processo de Software

- Três principais macro etapas:
  - **Requisitos**: o analista deve obter respostas a várias perguntas junto aos usuários.
    - Descreve as funcionalidades e recursos do software desde o que ele deve fazer até mesmo, se necessário, o que não deve fazer.
    - Perguntas chaves:
      - O que exatamente se espera que seja feito?
      - Qual a abrangência do software?
      - Quais os limites, ou o escopo do sistema?

Modelos de Processo de Software

- Três principais macro etapas:
  - **Projeto/Desenvolvimento**: o analista faz especificações técnicas detalhando a solução criada para atender ao usuário conforme os requisitos anteriores.
    - Geralmente, utiliza-se de diagramas UML.
    - Os programadores codificam os programas em alguma linguagem de programação.
    - Devem-se testar os programas exaustivamente para atingir um alto nível de qualidade, e após isso liberá-los para o uso.

Modelos de Processo de Software

- Três principais macro etapas:
  - **Implantação/Manutenção:** na implantação do software podem ocorrer vários problemas não previstos nas fases anteriores.
    - Momento de atualizar a versão do repositório e enviar para o usuário final.
    - A manutenção permanecerá durante toda sua vida útil.

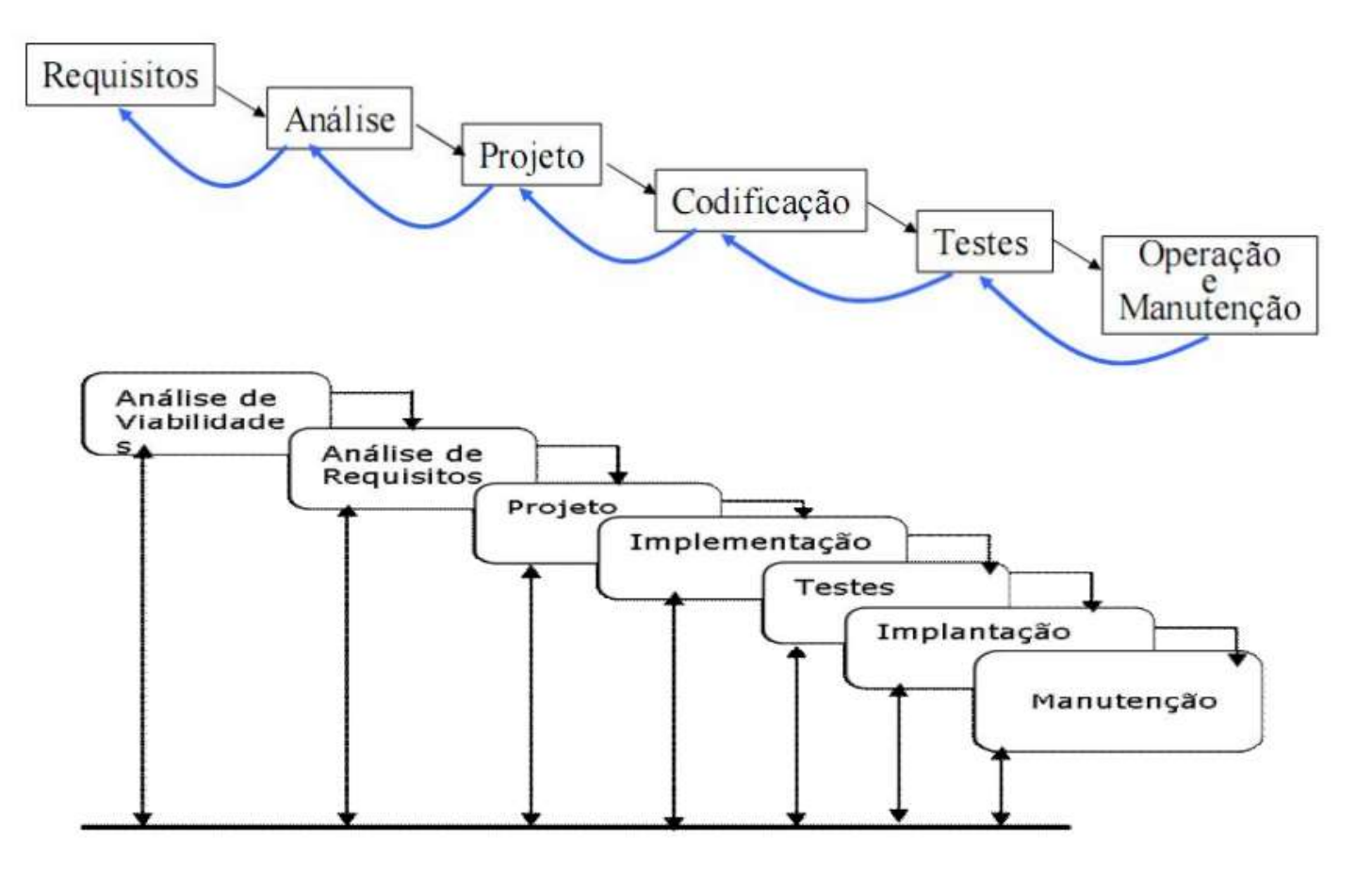
Modelo Balbúrdia

- Sistemas desenvolvidos na **informalidade** sem nenhum tipo de projeto ou documentação.
- Geralmente, em ciclo de apenas 2 fases: **implementação** e **implantação.**
- **Desordem** e **caos.**

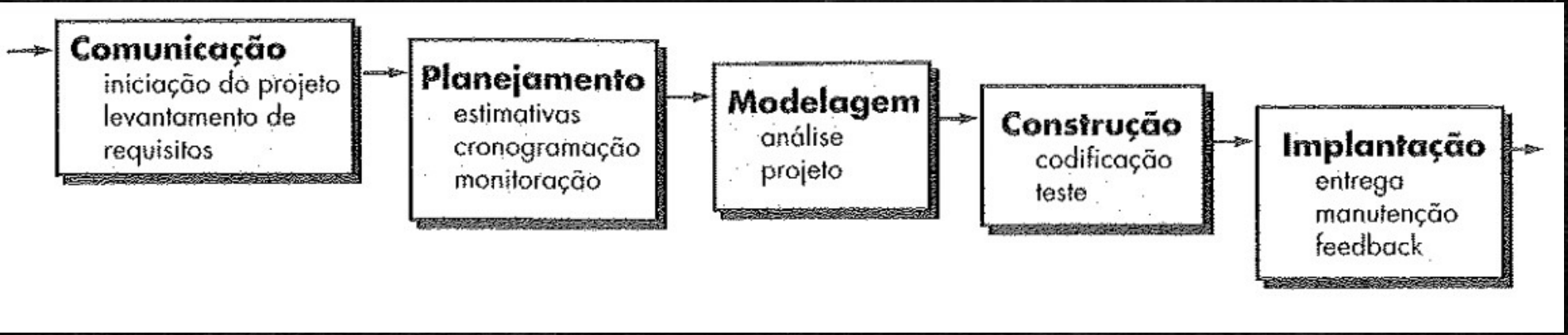
Modelo Cascata

- Aplicável quando os requisitos são **bem definidos** e **estáveis.**
- Ciclo de vida clássico.
- Atividades **sistemáticas** e **sequenciais.**
- Modelo **Linear.**

Modelo Cascata



Modelo Cascata



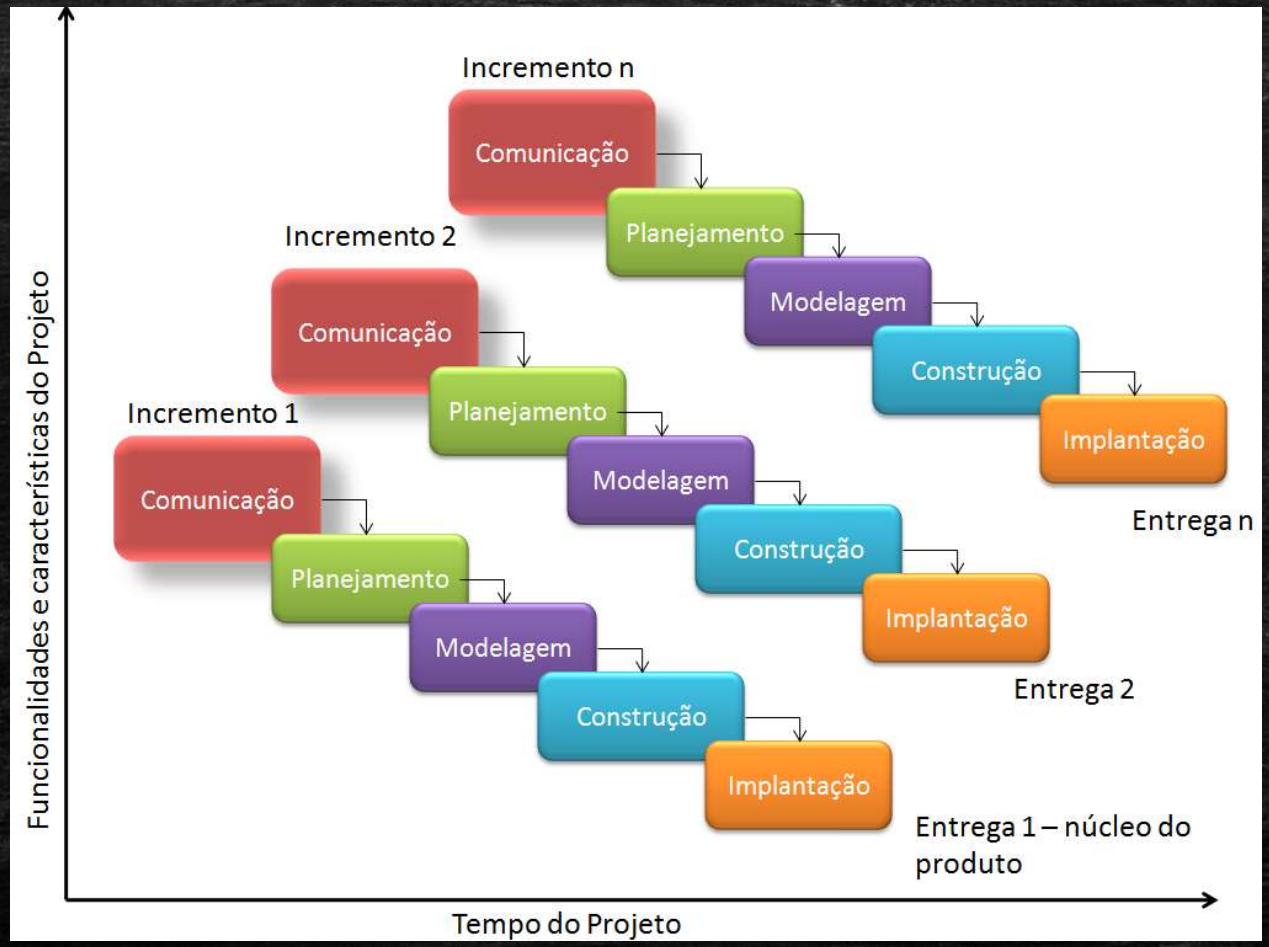
Modelos Incrementais

- Modelos Incrementais são desenvolvido com o conceito de **versões.**
- O sistema será especificado na documentação dos requisitos, e "**dividido**" **em** **subsistemas** por funcionalidades.
- Cada interação no ciclo do projeto pode gerar uma **versão.**
- Modelo cascata de maneira "interativa" e com entrega de **incrementos.**

Modelos Incrementais

- **Modelo Incremental:**
  - Entrega de incrementos constantes ao cliente.
  - Núcleo do produto é o primeiro incremento.
  - Cliente pode avaliar as entrega logo de inicio.

Modelos Incrementais

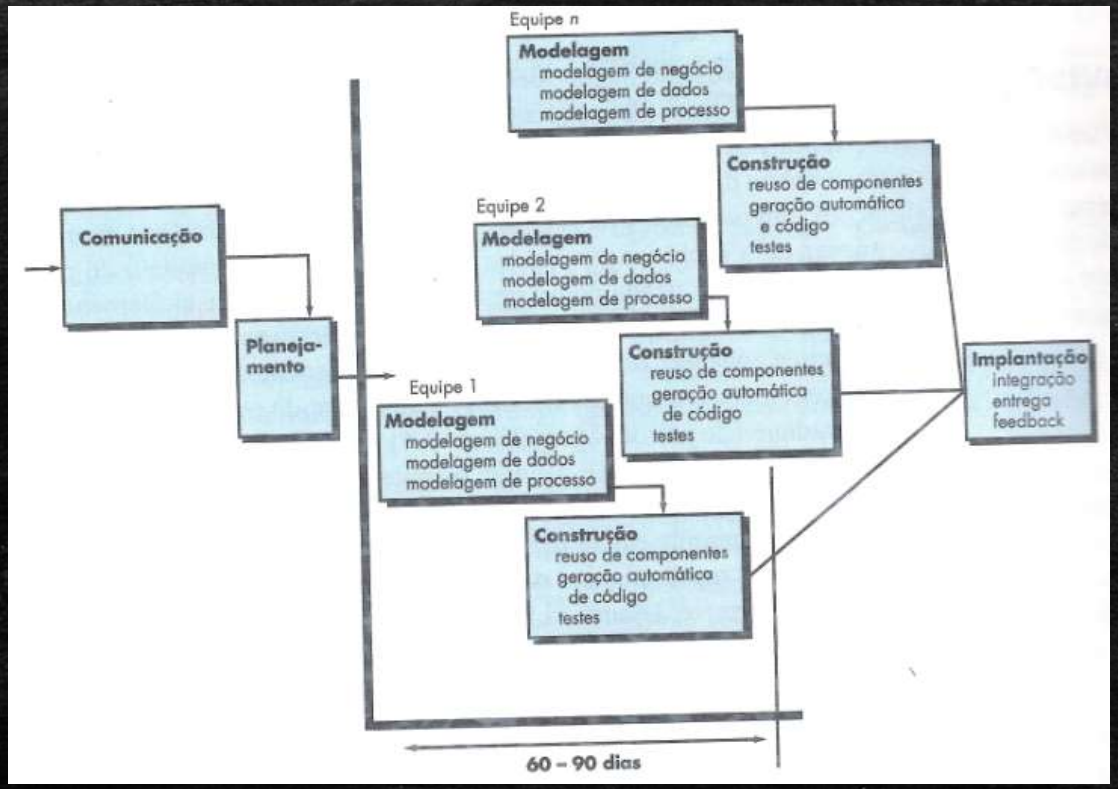


Modelos Incrementais

- **RAD (*Rapid Application Development*)**
  - Período de entrega curto (60 a 90 dias).
  - Maior numero de funcionários.
  - Muitas equipes.
  - Cada equipe trata uma função principal.
  - Desenvolvedores e clientes devem interagir constantemente.

Modelos Incrementais

- **RAD (*Rapid Application Development*)**



Modelos Evolucionários

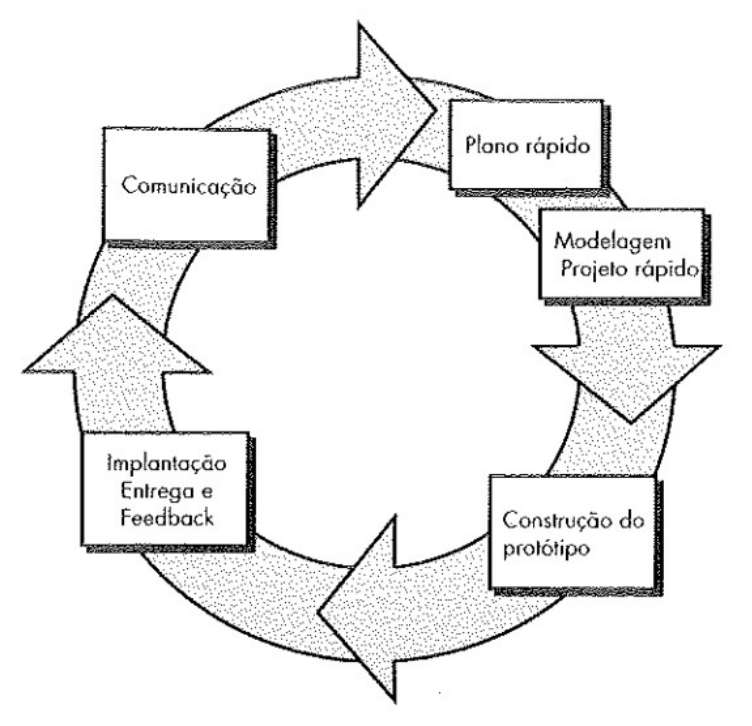
- Requisitos do negocio e do produto mudam frequentemente.
- Desenvolve versões cada vez mais completas do software, a cada iteração.
- Pode-se intervir e controlar o curso das atividades.
- Ex: **Prototipagem** e **Espiral**.

Modelos Evolucionários

- **Prototipagem**
  - É mais utilizado como uma **técnica** do que como modelo.
  - Sem preocupação na qualidade.
  - Recomendado quando os **requisitos não são bem compreendidos** pelos desenvolvedores.
  - Deve-se **descarta o protótipo**.
  - **Reduz os riscos e as incertezas** do desenvolvimento.

Modelos Evolucionários

- **Prototipagem**

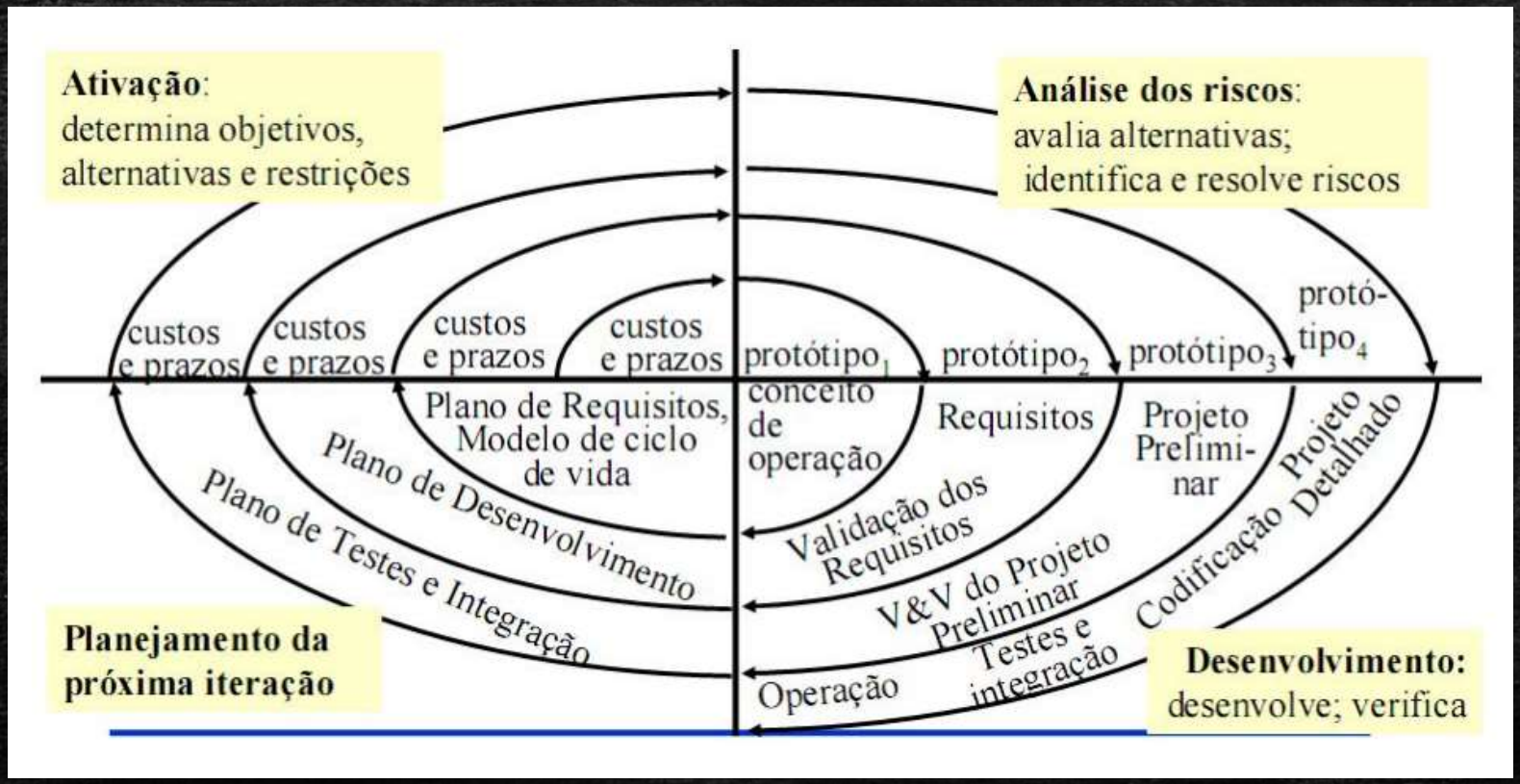


Modelos Evolucionários

- **Modelo Espiral**
  - Ideal em **sistemas mais complexos.**
  - Cada ciclo da espiral representa uma fase do processo de software.
  - Na **parte mais interna** relaciona-se o início da **visão da viabilidade** do sistema.
  - **Riscos** são **constantemente analisados** e **reduzidos.**
  - Gerência decide como estruturar o projeto em fases.

Modelos Evolucionários

- **Modelo Espiral**



Desenvolvimento Ágil

- *Manifesto for Agile Software Development* (Manifesto para Desenvolvimento Ágil de Software) criado em 2001 por Kent Beck, e mais 16 notáveis desenvolvedores (Aliança Ágil).
- Esse Manifesto ocorreu para ser um **contraponto as Metodologias** de Desenvolvimento **Prescritivas.**
- "Ser ágil é gerar **entregas contínuas**, **incrementais** e **frequentes de valor** para o cliente, focando em **equipes auto-organizadas.**" (Massari, 2016)

Desenvolvimento Ágil

- Entrega **incremental** e **avaliação** do cliente **constante.**
- **Equipes** de projeto **pequena** e **auto-organizadas.**
- As **atividades básicas** são reduzidas a um conjunto mínimo de tarefas.
- Valoriza-se mais:
  - **Indivíduos** e **interação** entre eles mais que processos e ferramentas.
  - **Software em funcionamento** mais que documentação abrangente.
  - **Colaboração com o cliente** mais que negociação de contratos.
  - **Responder a mudanças** mais que seguir um plano.

Desenvolvimento Ágil

helpper

Os 12 princípios ágeis

1. Priorize o Cliente
2. Mudanças são bem-vindas
3. Entregas que geram valor
4. Cooperação entre squads e stakeholders
5. Confie e apoie
6. Conversa face a face
7. Priorize o que funciona
8. Desenvolvimento incremental
9. Exelência técnica
10. Mantenha a simplicidade
11. Times autônomos e auto-gerenciavéis
12. Retrospectiva e planejamento

Processo Ágil

- Suposições chaves para utilizar um processo ágil:
  - A **dificuldade de prever** antecipadamente quais **requisitos** de softwares vão persistir e quais serão modificados.
  - A **dificuldade de prever** quanto de **projeto** é necessário antes que a construção seja usada para comprovar o projeto.
  - Análise, projeto, construção e teste **não são previsíveis** como é desejado.

Processo Ágil

- Um processo ágil deve ser **adaptável incrementalmente**.
- Os processos de especificação, projeto e implementação são intercalados.
- Interfaces de Usuário do sistema são criadas com sistemas interativos ou bibliotecas que permitem a fácil construção.

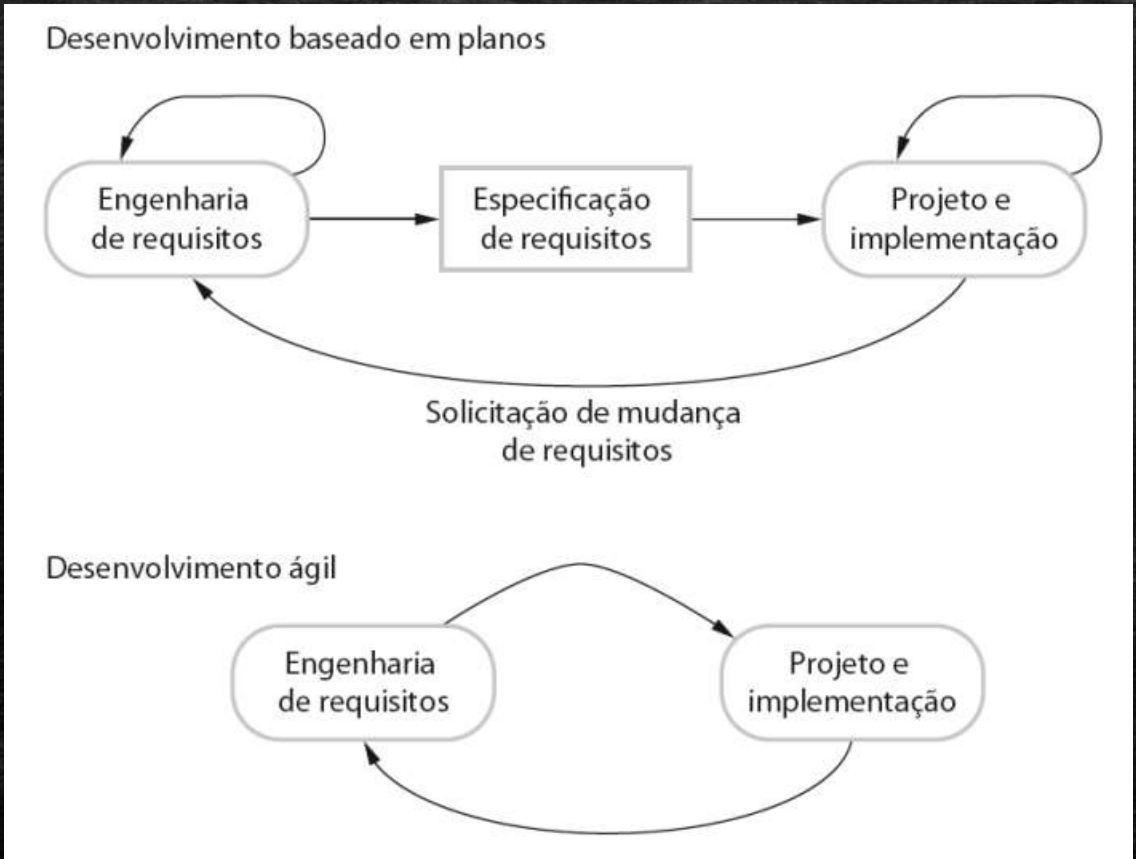
Processo Ágil

- O processo se molda às necessidades das pessoas e da equipe.
- Principais características de uma equipe de Desenvolvimento Ágil:
  - **Competência.**
  - **Foco comum.**
  - **Colaboração.**
  - **Capacidade de tomada de decisão.**
  - **Habilidade de resolver problemas vagos.**
  - **Respeito e confiança mútua.**
  - **Auto-organização.**

Desenvolvimento ágil e dirigido a planos

- **Dirigido a planos (tradicional)**
  - é baseada em estágios de desenvolvimento separados, com os produtos a serem produzidos em cada um desses estágios planejados antecipadamente.
- **Ágil**
  - Especificação, projeto, implementação e teste são intercalados e os produtos do processo de desenvolvimento são decididos através de um processo de negociação, durante processo de desenvolvimento do software.

Desenvolvimento ágil e dirigido a planos

- 

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - É importante ter uma **especificação e projeto bem detalhados** antes de passar para a implementação?
    - Caso seja, provavelmente você precisa usar uma abordagem **dirigida a planos**.

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - Uma estratégia de **entrega incremental** onde você entrega os software para os clientes e **recebe feedback rápido** deles é possível?
    - Caso seja, considere usar **métodos ágeis.**

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - Qual o **tamanho do sistema** a ser desenvolvido?
    - Os **métodos ágeis** são mais efetivos quando o sistema pode ser desenvolvido com uma **equipe pequena** que pode se **comunicar informalmente.**
    - O que pode não ser possível para **sistemas grandes** que **requerem grande equipes** de desenvolvimento, nesses casos, deve ser usada uma **abordagem dirigida a planos.**

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - Que **tipo de sistema** está sendo desenvolvido?
    - Abordagens **dirigidas a planos** podem ser necessárias para sistemas que **requerem muita análise** antes da implementação.
    - ex: **sistema** que opere **em tempo real** com requisitos de temporização complexos.

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - Qual é o **tempo de vida esperado** para o sistema?
    - Sistemas com **longo tempo de vida** podem precisar de **mais documentação** de projeto para comunicar as intenções originais dos desenvolvedores do sistema para a equipe de suporte.

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - Quais **tecnologias estão disponíveis** para manter o desenvolvimento do sistema?
    - **Métodos ágeis** dependem de **boas ferramentas** para acompanhar um sistema em evolução.

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - Como está **organizada a equipe** de desenvolvimento?
    - Se a **equipe** de desenvolvimento **está distribuída** ou se parte do **desenvolvimento está sendo terceirizado** você pode precisar desenvolver **documentos de projeto** para que haja comunicação entre as equipes de desenvolvimento, ou seja, abordagem dirigida a planos.

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - Existem **questões culturais ou organizacionais** que podem afetar o desenvolvimento do sistema?
    - As **organizações tradicionais** de engenharia têm uma cultura de **desenvolvimento dirigido a planos**, o que é padrão em engenharia.

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - O **quão bons são** os **projetistas** e os **programadores** da equipe de desenvolvimento?
    - É dito que os **métodos ágeis** requerem um **nível de habilidade mais alto** do que as abordagens dirigidas a planos, nas quais os **programadores** simplesmente **traduzem** um **projeto** detalhado em **código**.

Desenvolvimento ágil e dirigido a planos

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:
  - O sistema está sujeito a **regulamentação externa**?
    - Se o sistema precisa ser aprovado por um **regulador externo** então provavelmente requisitaram a você a produção de **documentação detalhada** como parte da documentação de segurança do sistema código.
    - ex: A Federal Aviation Administration aprova softwares críticos para a operação de um avião.

Modelos Ágeis

- Extreme Programming – XP
- Scrum

Extreme Programming – XP

- O modelo mais amplamente utilizado.
- Desenvolvido por Kent Beck, em 1999.
- Paradigma, predileto, é o **orientado a objetos.**
- As atividades de arcabouço são: **Planejamento, Projeto, Codificação e Teste.**

.

Extreme Programming – XP

- Existe uma grande ênfase ao trabalho em **duplas**, no qual um analista mais experiente trabalha com um novato.
- **Requisitos** do usuário são **expressos com cenários** ou **histórias dos usuários.**
  - Esses são escritos em cartões e a equipe de desenvolvimento os divide em tarefas de implementação.

Extreme Programming – XP

História de prescrição de medicamentos

Prescrição de medicamentos

Kate é uma médica que deseja prescrever medicamentos para um paciente de uma clínica. O prontuário do paciente já está sendo exibido em seu computador, assim, ela clica o campo 'medicação' e pode selecionar 'medicação atual', 'nova medicação', ou 'formulário'.

Se ela selecionar 'medicação atual', o sistema pede que ela verifique a dose. Se ela quiser mudar a dose, ela altera esta e em seguida, confirma a prescrição.

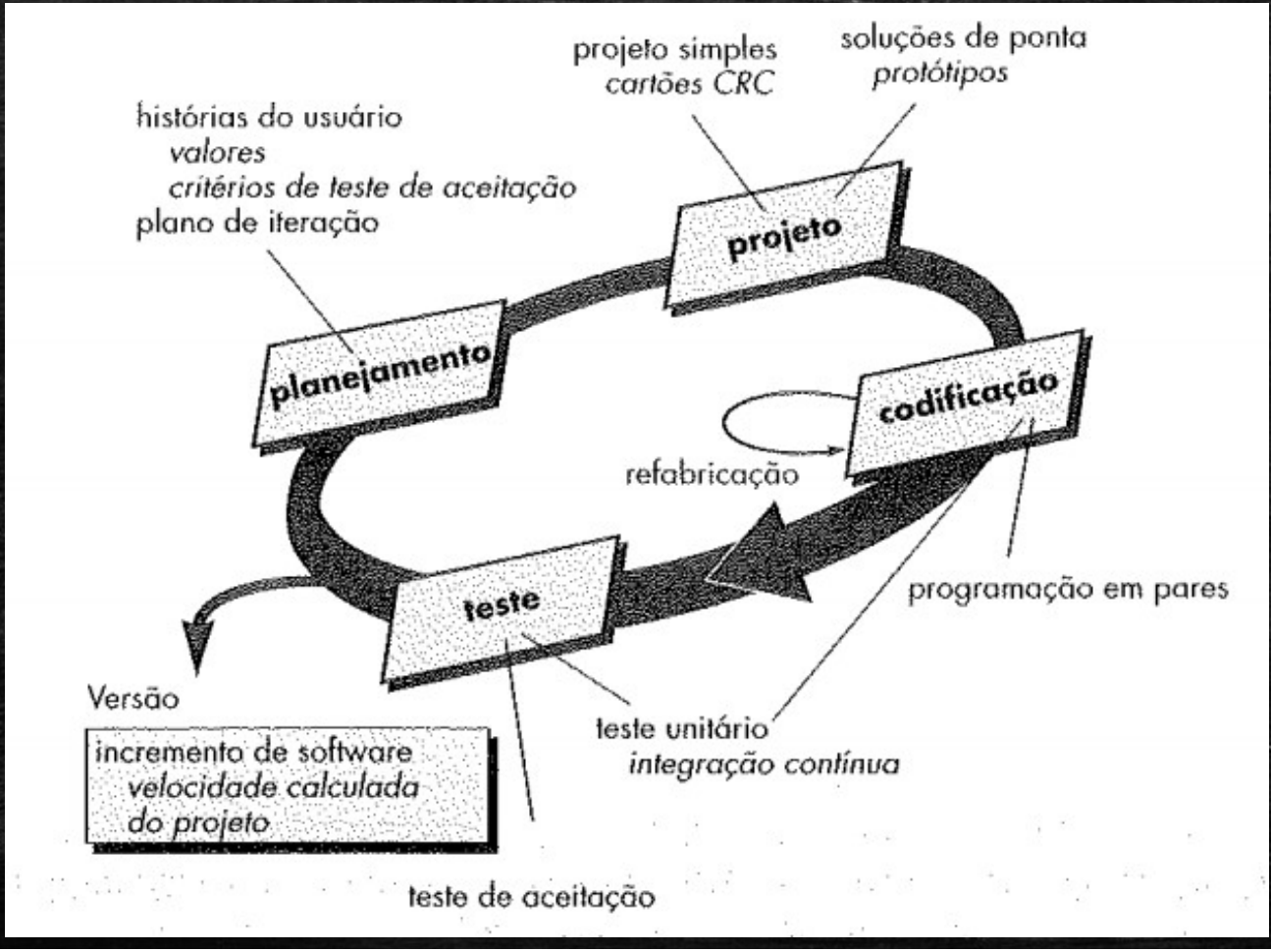
Se ela escolher 'nova medicação', o sistema assume que ela sabe qual medicação receitar.
Ela digita as primeiras letras do nome do medicamento. O sistema exibe uma lista de possíveis fármacos que começam com essas letras. Ela escolhe a medicação requerida e o sistema responde, pedindo-lhe para verificar se o medicamento selecionado está correto.
Ela insere a dose e, em seguida, confirma a prescrição.

Se ela escolhe 'formulário', o sistema exibe uma caixa de busca para o formulário aprovado.
Ela pode, então, procurar pelo medicamento requerido. Ela seleciona um medicamento e é solicitado que verifique se a medicação está correta. Ela insere a dose e, em seguida, confirma a prescrição.

O sistema sempre verifica se a dose está dentro da faixa permitida. Caso não esteja, Kate é convidada a alterar a dose.

Após Kate confirmar a prescrição, esta será exibida para verificação. Ela pode escolher 'OK' ou 'Alterar'. Se clicar em 'OK', a prescrição fica gravada nos bancos de dados da auditoria.
Se ela clicar em 'Alterar', reinicia o processo de 'Prescrição de Medicamentos'.

Extreme Programming – XP



Extreme Programming – XP

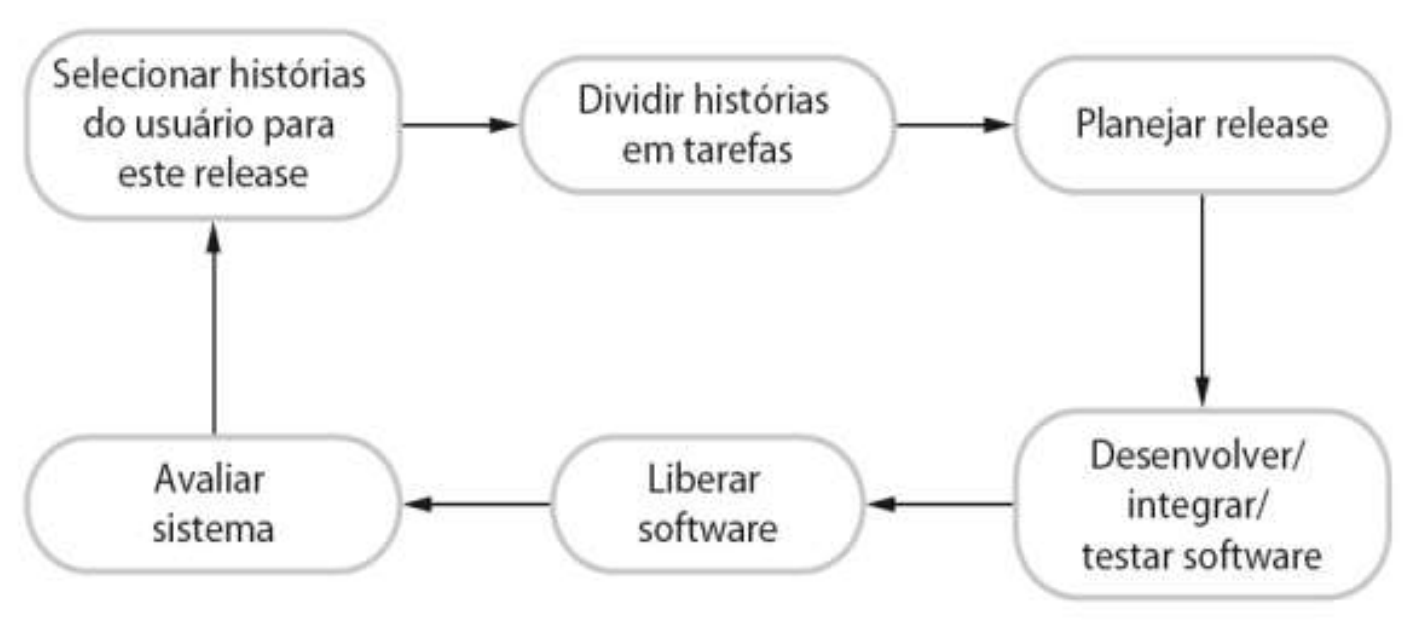
Cartões CRC (Class Responsibility Card): é um cartão que descreve uma classe através de suas responsabilidades e colaboradores.

Classes associadas

	Classe: Conta Corrente	
	Responsabilidade	**Colaboração**
atributos	Saber o seu saldo	Cliente
	Saber seu cliente	Histórico de Transações
	Saber seu número	
métodos	Manter histórico de transações	
	Realizar saques e depósitos	

Extreme Programming – XP

O ciclo de um release em Extreme Programming



Extreme Programming – XP

Princípio ou prática	Descrição
Planejamento incremental	Os requisitos são gravados em cartões de estória e as estórias que serão incluídas em um release são determinadas pelo tempo disponível e sua relativa prioridade. Os desenvolvedores dividem essas estórias em 'Tarefas'.
Pequenos *releases*	Em primeiro lugar, desenvolve-se um conjunto mínimo de funcionalidades útil, que fornece o valor do negócio. *Releases* do sistema são frequentes e gradualmente adicionam funcionalidade ao primeiro *release*.
Projeto simples	Cada projeto é realizado para atender às necessidades atuais, e nada mais.

Extreme Programming – XP

Princípio ou prática	Descrição
Desenvolvimento *test-first*	Um *framework* de testes iniciais automatizados é usado para escrever os testes para uma nova funcionalidade antes que a funcionalidade em si seja implementada.
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente assim que encontrarem melhorias de código. Isso mantém o código simples e manutenível.
Programação em pares	Os desenvolvedores trabalham em pares, verificando o trabalho dos outros e prestando apoio para um bom trabalho sempre.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo que não se desenvolvam ilhas de *expertise*. Todos os conhecimentos e todos os desenvolvedores assumem responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.

Extreme Programming – XP

Princípio ou prática	Descrição
Integração contínua	Assim que o trabalho em uma tarefa é concluído, ele é integrado ao sistema como um todo. Após essa integração, todos os testes de unidade do sistema devem passar.
Ritmo sustentável	Grandes quantidades de horas-extra não são consideradas aceitáveis, pois o resultado final, muitas vezes, é a redução da qualidade do código e da produtividade a médio prazo.
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível todo o tempo à equipe de XP. Em um processo de Extreme Programming, o cliente é um membro da equipe de desenvolvimento e é responsável por levar a ela os requisitos de sistema para implementação.

Extreme Programming – XP

Descrição de caso de teste para verificação de dose

Teste 4: Verificação de dose

Entrada:
1. Um número em mg representando uma única dose da medicação.
2. Um número que representa o número de doses únicas por dia.

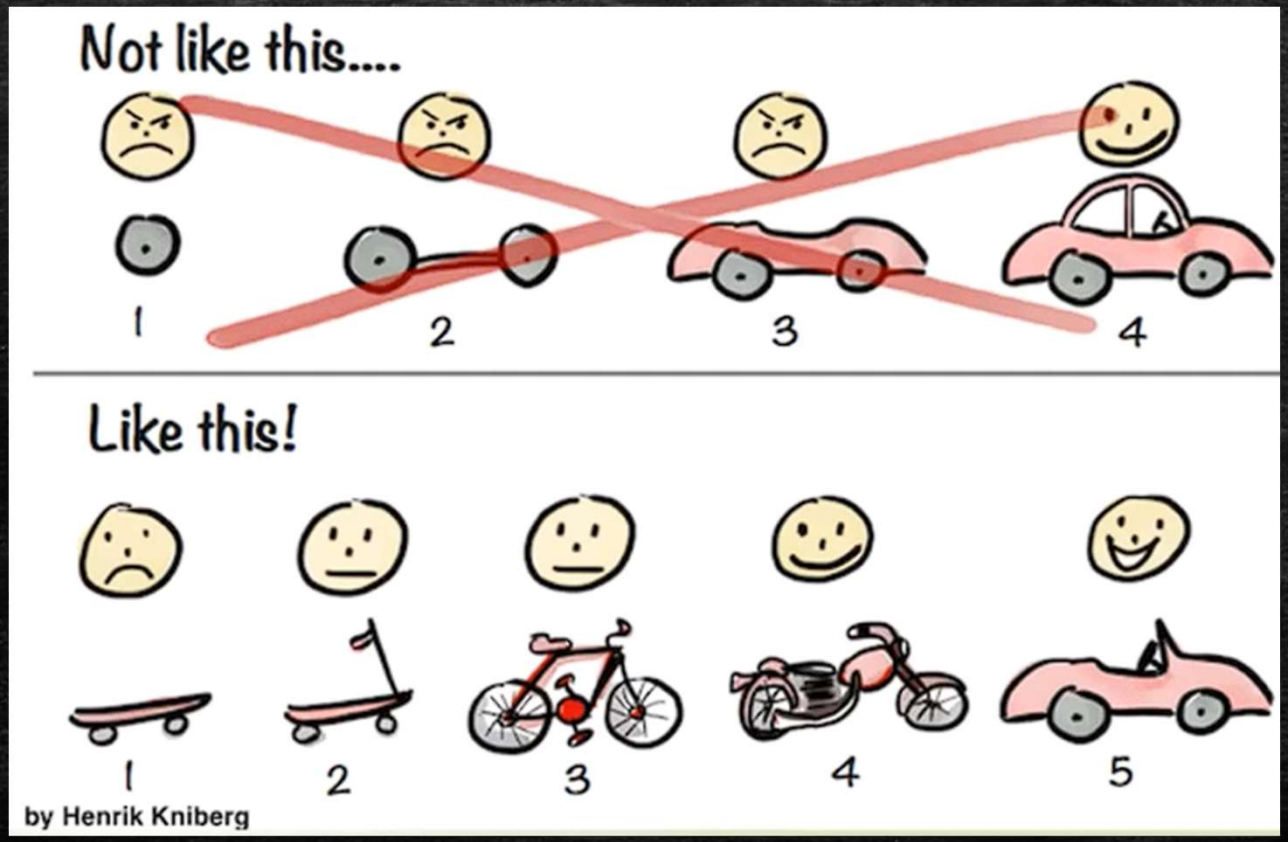
Testes:
1. Teste para entradas em que a dose única é correta, mas a frequência é muito alta.
2. Teste para entradas em que a única dose é muito alta e muito baixa.
3. Teste para entradas em que a dose única x frequência é muito alta e muito baixa.
4. Teste para entradas em que a dose única x frequência é permitida.

Saída:
Mensagem de OK ou erro indicando que a dose está fora da faixa de segurança.

Scrum

- Ken Schwaber e Jeff Sutherland primeiramente co-apresentaram o Scrum na Conferência OOPSLA em 1995.
- É um **framework leve** que ajuda pessoas, times e organizações a **gerar valor** por meio de soluções **adaptativas** para problemas complexos.
- Os pilares do Scrum são: **transparência, inspeção e adaptação.**
  - Valores do Scrum: **Compromisso, Foco, Abertura, Respeito e Coragem.**

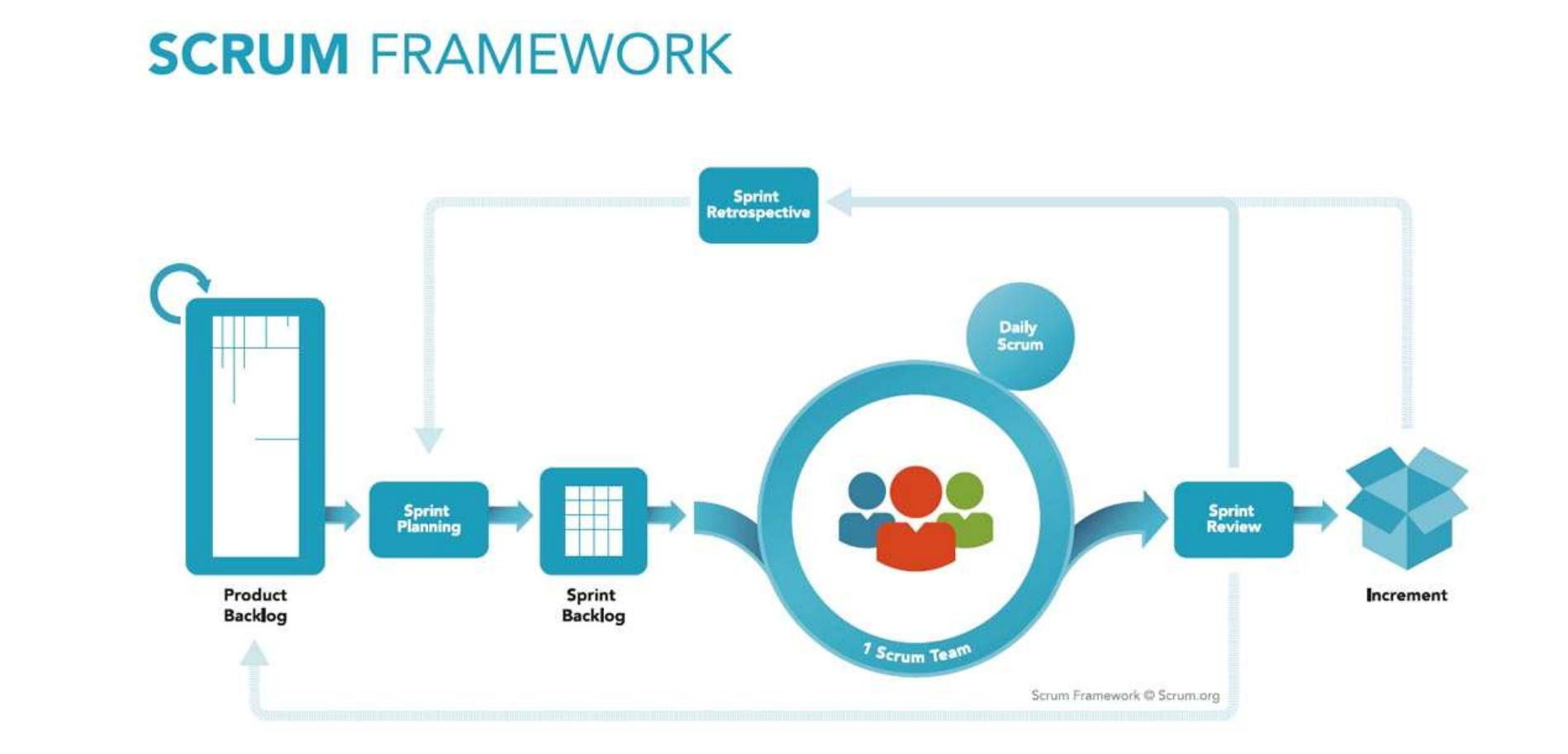
Scrum



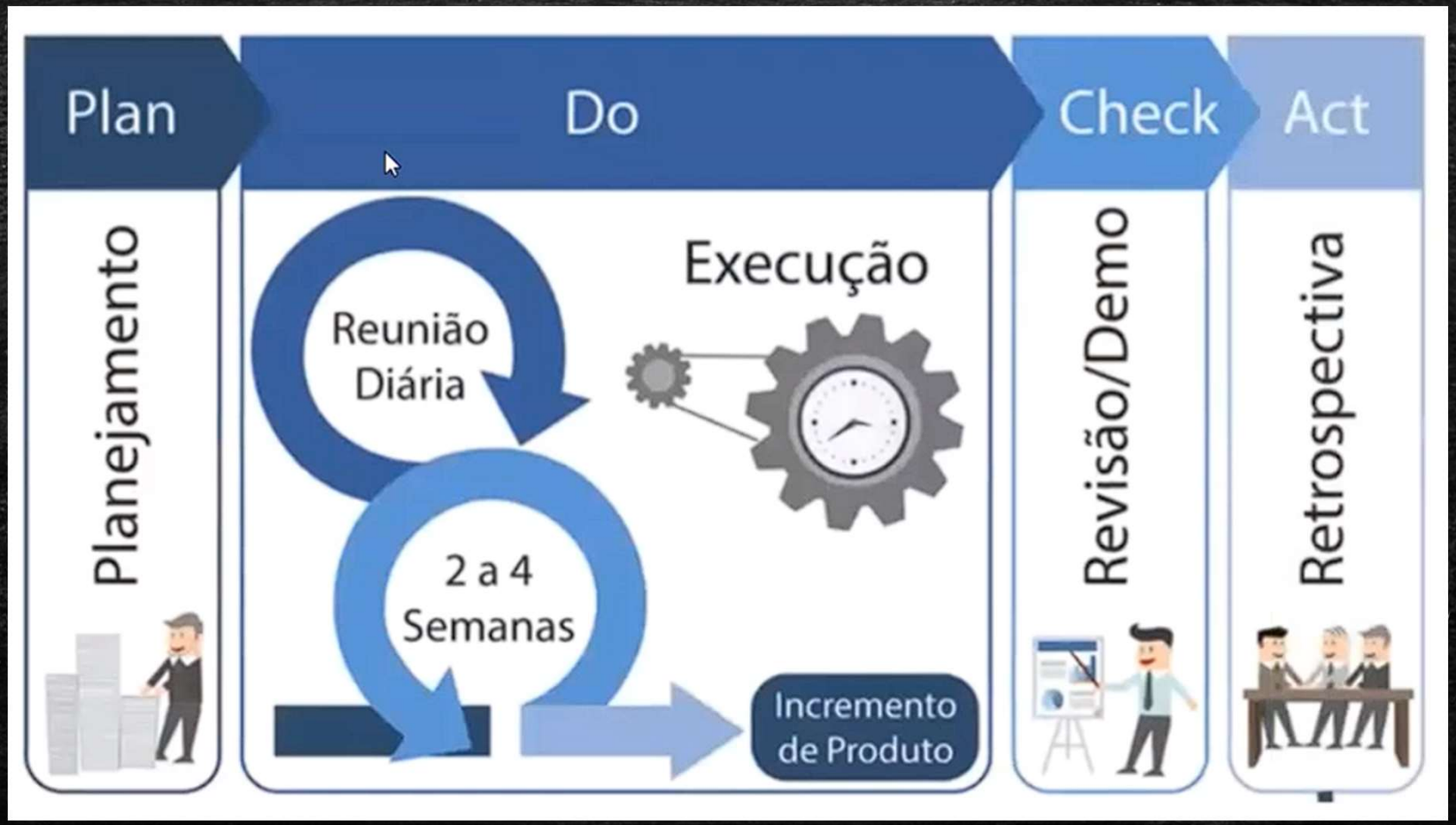
Scrum

- Scrum é baseado no **empirismo** e ***lean thinking***.
  - A teoria do empirismo afirma que o conhecimento vem da experiência e da tomada de decisões com base no que é observado.
  - O *lean thinking é uma filosofia de gestão que* reduz o desperdício e se concentra no essencial.
- Scrum emprega uma abordagem **iterativa e incremental** para otimizar a previsibilidade e controlar o risco.
- Vários processos, técnicas e métodos podem ser empregados com o framework.

Scrum

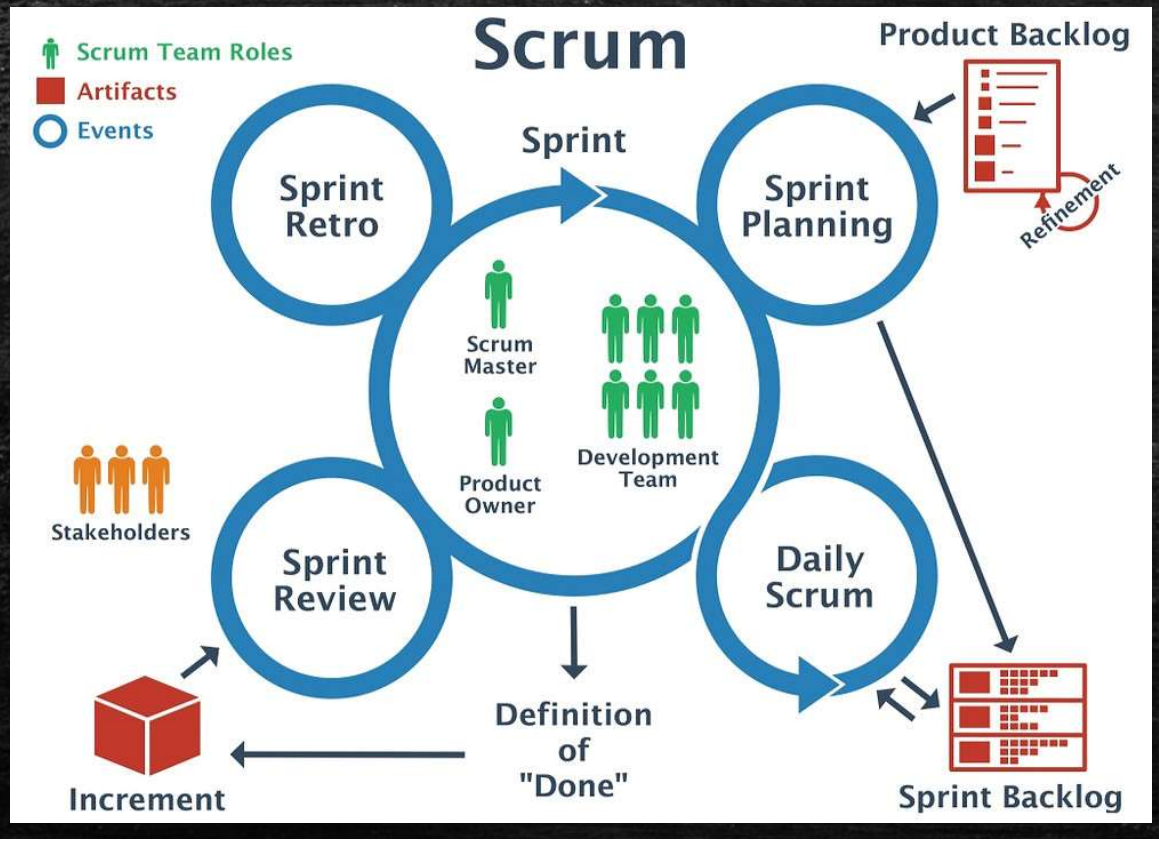

SCRUM FRAMEWORK
Sprint Retrospective
Daily Scrum
Product Backlog
Sprint Planning
Sprint Backlog
1 Scrum Team
Sprint Review
Increment
Scrum Framework © Scrum.org

Scrum


Plan
Do
Check
Act
Planejamento
Reunião Diária
Execução
2 a 4 Semanas
Incremento de Produto
Revisão/Demo
Retrospectiva

Scrum

- Papéis, Artefatos e Cerimônias

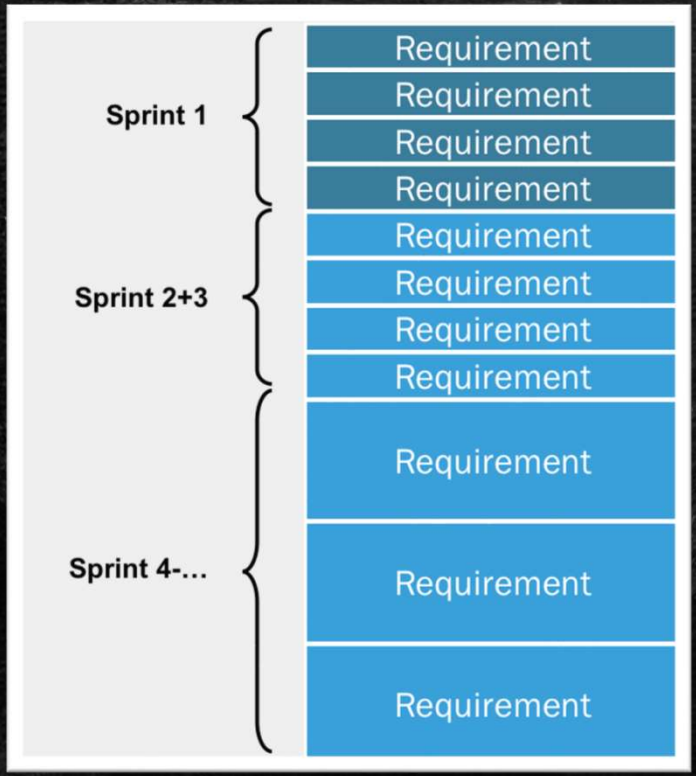


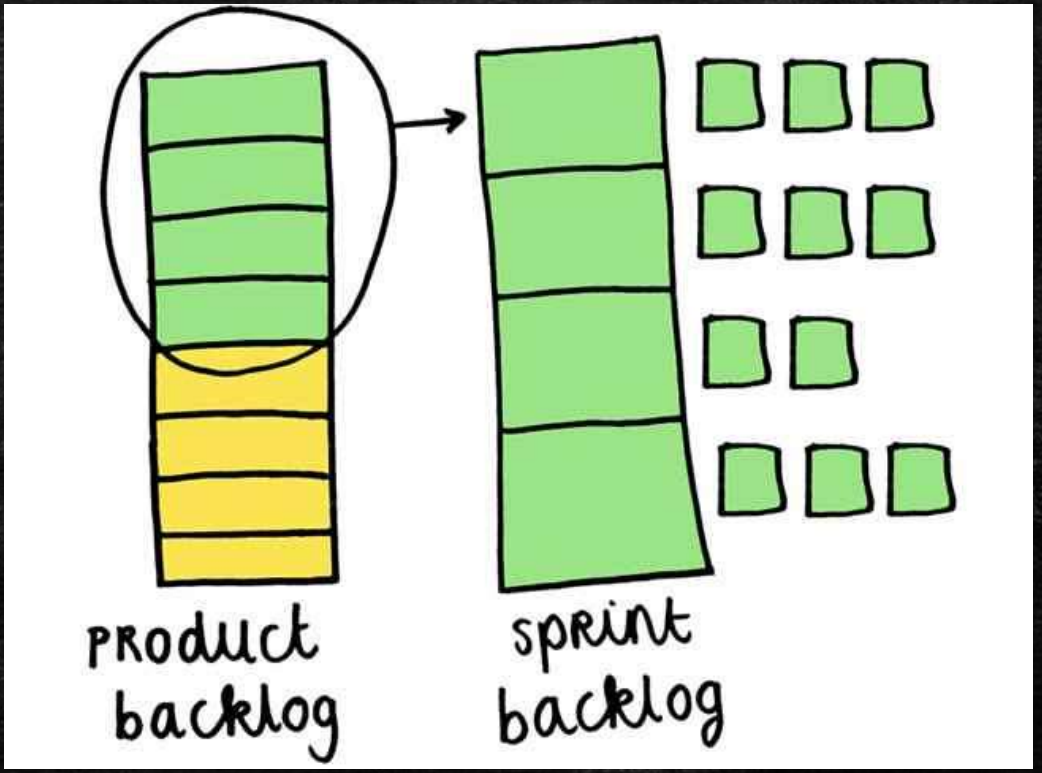
O que é Product Backlog?
Refere-se a uma **lista ordenada de pedidos** a serem realizadas num determinado **prazo** e que no final gerará o produto desejado.

O que é Sprint Backlog?
É uma **lista de tarefas** referentes algum parte do PB que o **Scrum Team** se compromete a fazer **em um ciclo** da Sprint.

Scrum

- Product Backlog x Spring Backlog





Scrum

- Product Backlog x Spring Backlog

Product Backlog	Spring Backlog
PO tem o controle sobre o PB	Proprietário é Equipe de Desenvolvimento
Focado em toda a visão do produto	Focado em cumprir a meta do sprint
Sem limite de tempo para completar todo o Backlog	Concluído em um período de tempo designado
Muda até a conclusão do produto	Um subconjunto do backlog do produto com tarefas definidas

Scrum

- ***Sprint***: é uma **interação** e um **evento** com duração de no máximo 1 mês.
  - Possui uma **meta bem estabelecida** com objetivo claro.
  - A Sprint só **ocorre após** um evento de planejamento, chamada de **Sprint Planning.**
    - Nessa reunião, definem-se os itens a serem realizados durante o ciclo da Sprint e que são retirados do Product Backlog (PB).
    - Os itens da sprint compõem o que é chamado de Sprint Backlog.
      - Nesse momento, estes itens devem ser refinados e expandidos pelo time de desenvolvimento, com as informações técnicas e o tempo necessários para a conclusão de cada item.

Scrum

- ***Product Owner***: conhece o valor de negócio do produto, recebe as demandas dos clientes (*stakeholders*) e entende o que é esse produto.
  - Realiza o **levantamento** de todo o **Backlog do produto.**
  - **Define prioridades** dentro do Backlog e **de que forma** o Time Scrum **entregará valor** a cada sprint.

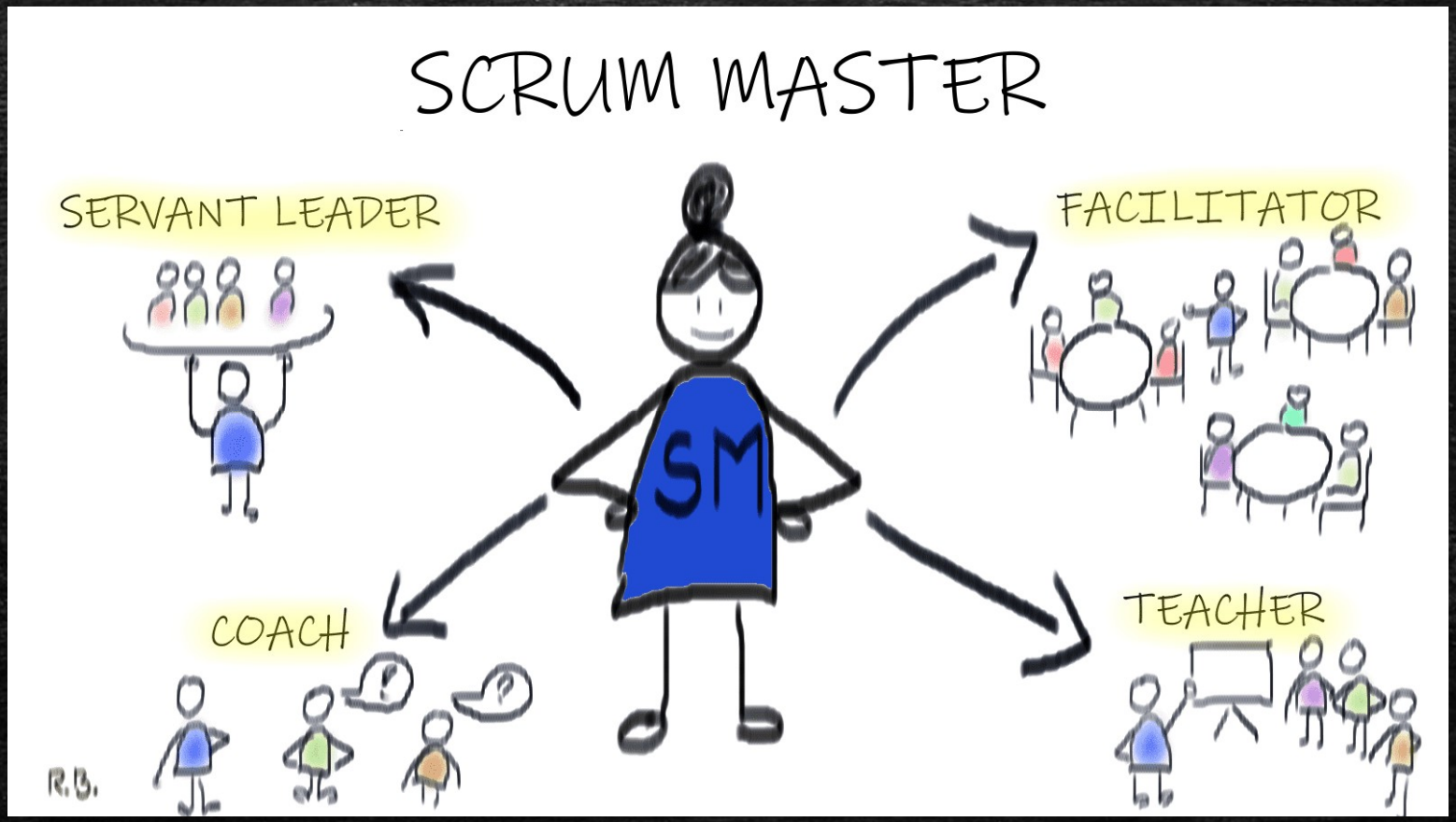
Scrum

PRODUCT OWNER:
PRINCIPAIS FUNÇÕES
Definir a visão
Gerenciar o backlog do produto
Priorizar as necessidades
Supervisionar os estágios de desenvolvimento
Antecipar as necessidades do cliente
Atuar como ligação principal
Avaliar o progresso do produto
Participar da Daily semanal
Encerrar a Sprint

Scrum

- ***Scrum Master***: Tem o papel de **proteger a equipe** de desenvolvimento de distrações externas.
  - É um **facilitador**, que **organiza reuniões** diárias, **controla o backlog**, **registra decisões**, **mede o progresso** comparado o backlog e **se comunica** com os **clientes** e a **gerencia** externa à equipe.
  - É o *coach* do Time Scrum e garante todo o **suporte ferramental e motivacional** para que a solução seja construída.

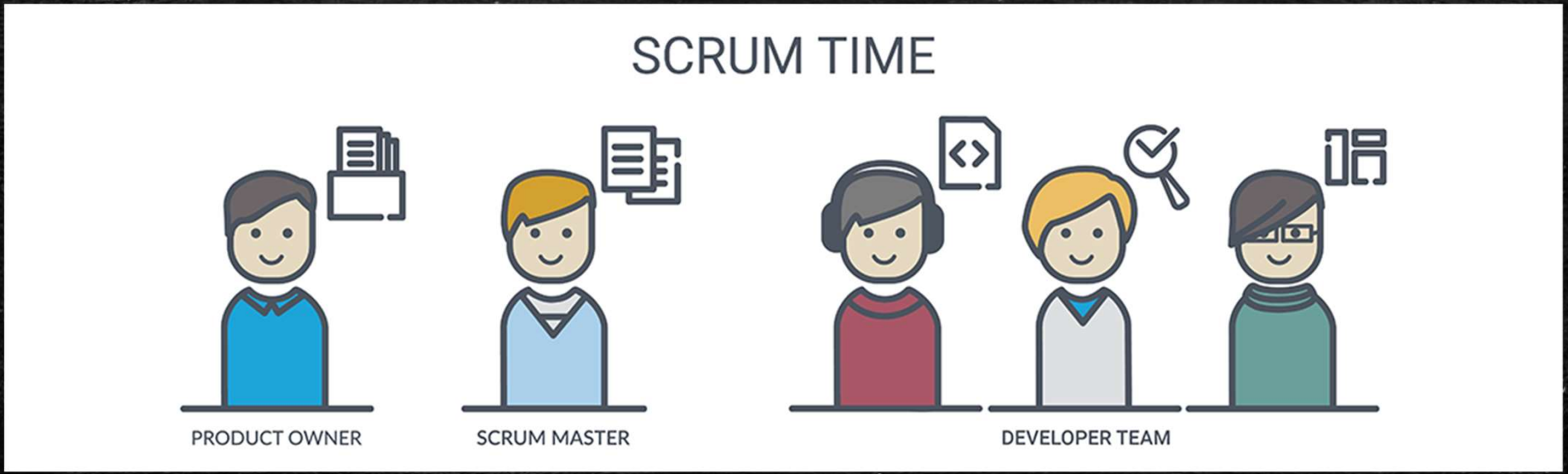
Scrum


SCRUM MASTER
SERVANT LEADER
FACILITATOR
SM
COACH
TEACHER
R.B.

Scrum

- ***Scrum Team*** (Time Scrum): é **auto-gerenciáveis** e **multifuncional**, escolhendo qual é a sua melhor forma de trabalhar e sendo capaz de completar todo o trabalho sem depender de outros externos.
  - **Determina** sua própria **velocidade** de trabalho, bem como, sua **capacidade**.
  - É a **união de todos os papeis** do scrum, ou seja, PO, SM e os demais membros da equipe.

Scrum



Scrum

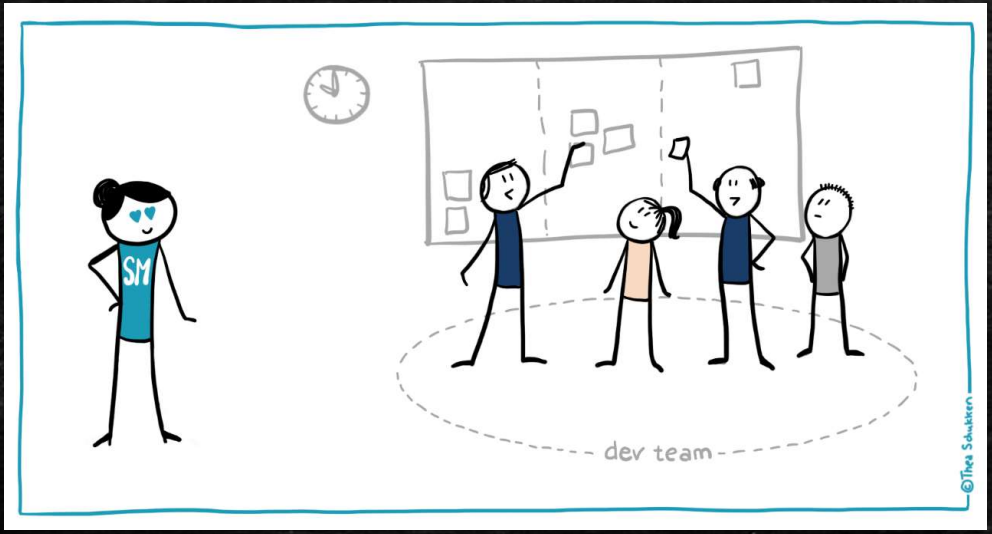
- Crie uma equipe entre 4 e 6 pessoas.
  - Escolha 1 para ser o Product Owner.
  - Escolha 1 para ser o Scrum Master.
  - Escolha um nome para a equipe.

Scrum

- ***Sprint Planning***: inicia a Sprint ao **definir o trabalho a ser realizado** na Sprint.
  - Este plano resultante é criado pelo trabalho colaborativo de todo o Scrum Team.
  - O Product Owner garante que os participantes estejam preparados para discutir os itens mais importantes do Product Backlog e como eles são mapeados para a meta do produto.
  - Tópicos:
    - **Por que** esta Sprint é valiosa?
    - **O que** pode ser feito nesta Sprint?
    - **Como** o trabalho escolhido será realizado?

Scrum

- ***Daily Scrum*** (Reuniões diárias): São reuniões que ocorrem diariamente durando cerca de 15 minutos.
  - O que eu fiz ontem?
  - O que eu farei hoje?
  - Eu tenho algum obstáculo?

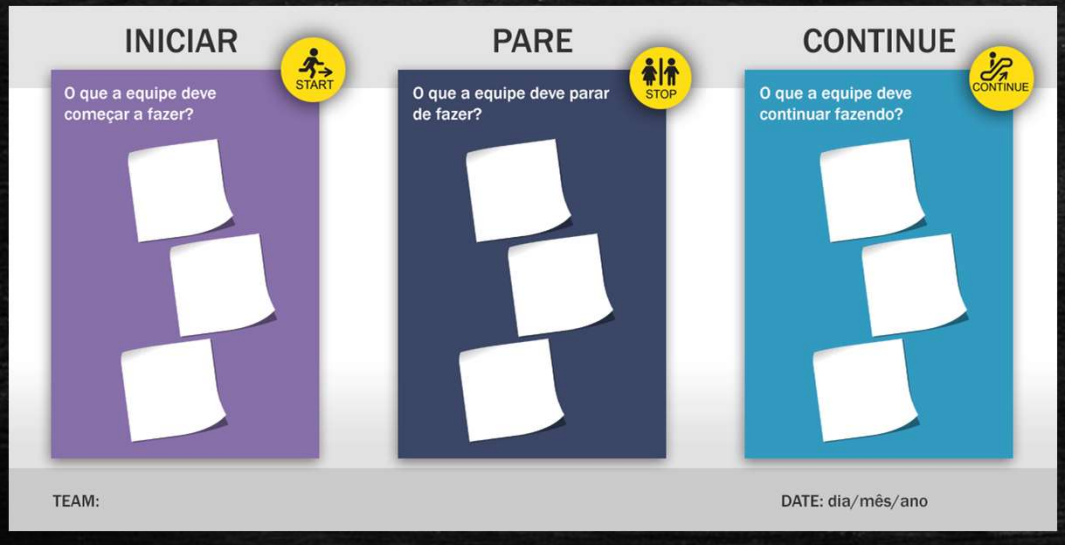


Scrum

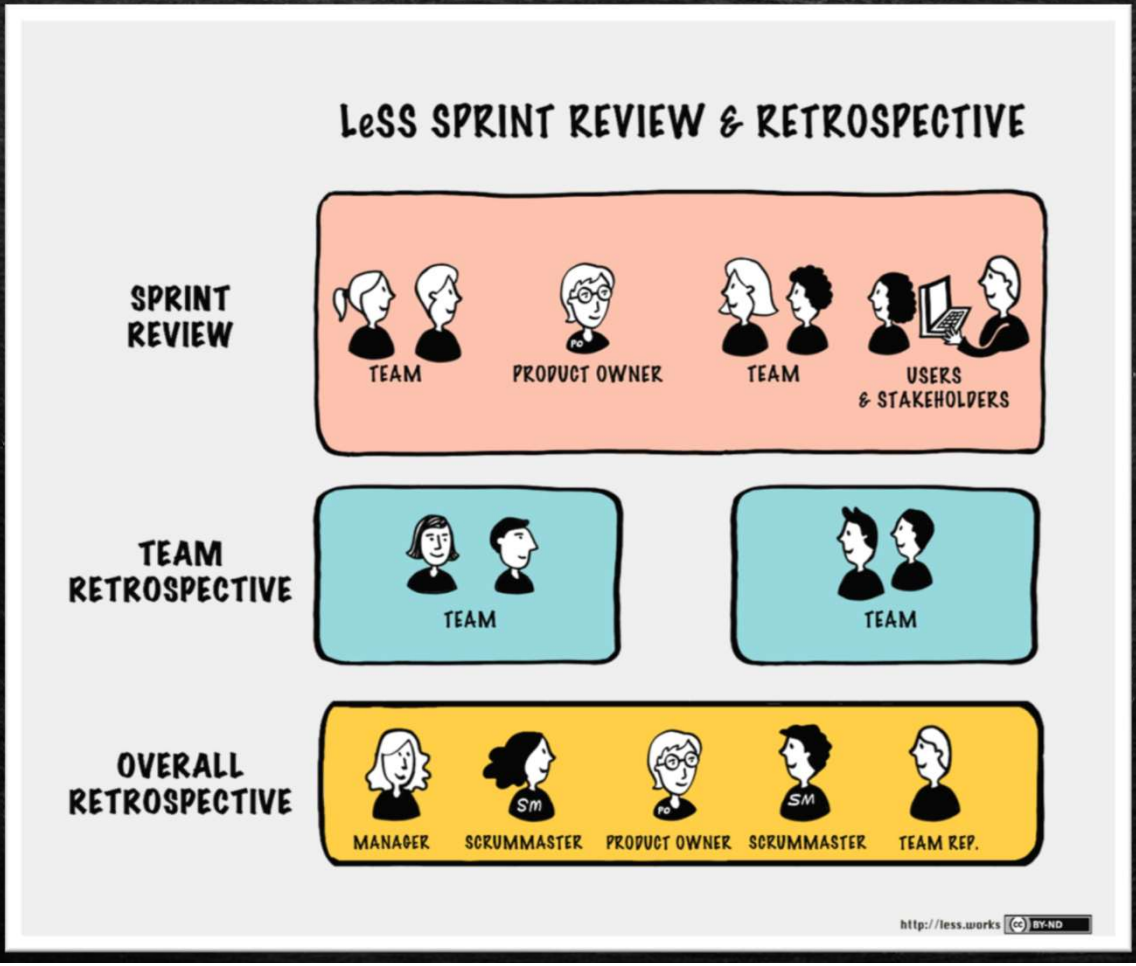
- ***Sprint Review*** (Revisão Sprint): é um **ponto de inspeção** e **adaptação** que ocorre no **final de uma sprint.**
  - *Stakeholders* se reúnem e verificam o que foi construído, discutindo mudanças e novos apontamentos em conjunto com time scrum.

Scrum

- ***Sprint Retro*** (Retrospectica da Sprint): funciona como uma **cerimônia para avaliar** como ocorreu o **processo da sprint.**
  - Faz-se uma retrospectiva verificando o que foi bom e o que deve ser melhorado e então realiza-se um plano buscando uma melhoria continua do processo realizado.



Scrum

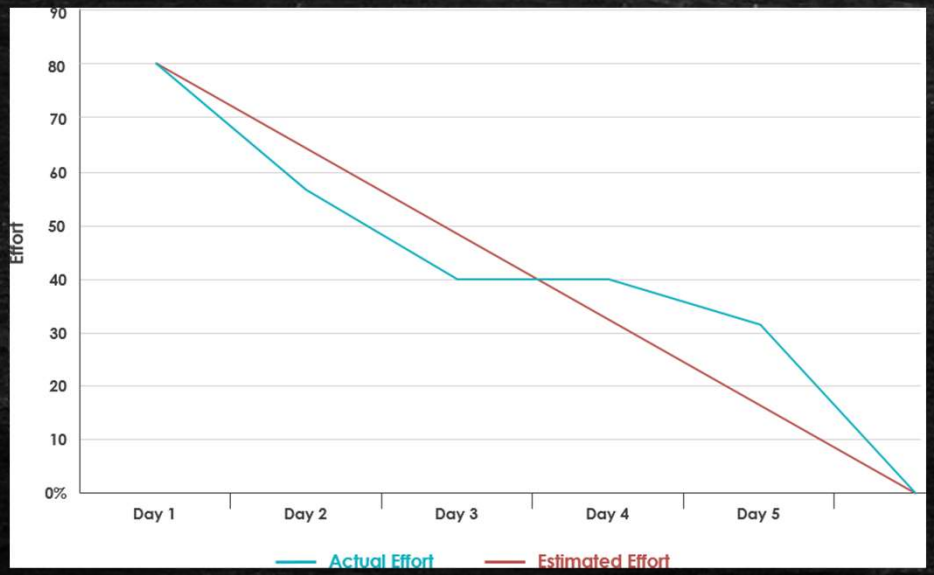

LeSS SPRINT REVIEW & RETROSPECTIVE
SPRINT REVIEW
TEAM
PRODUCT OWNER
TEAM
USERS & STAKEHOLDERS
TEAM RETROSPECTIVE
TEAM
TEAM
OVERALL RETROSPECTIVE
MANAGER
SCRUMMASTER
PRODUCT OWNER
SCRUMMASTER
TEAM REP.
http://less.works
BY-ND

Scrum

- ***Scrum Artifacts*** (Artefatos do Scrum): **representam trabalho ou valor.**
  - São projetados para maximizar a transparência das principais informações.
  - Cada artefato contém um compromisso e o foco contra o qual o progresso pode ser medido:
    - Para o *Product Backlog*, é a Meta do produto.
    - Para o *Sprint Backlog*, é a Meta da Sprint.
    - Para o incremento, é a Definição de Pronto.

Scrum

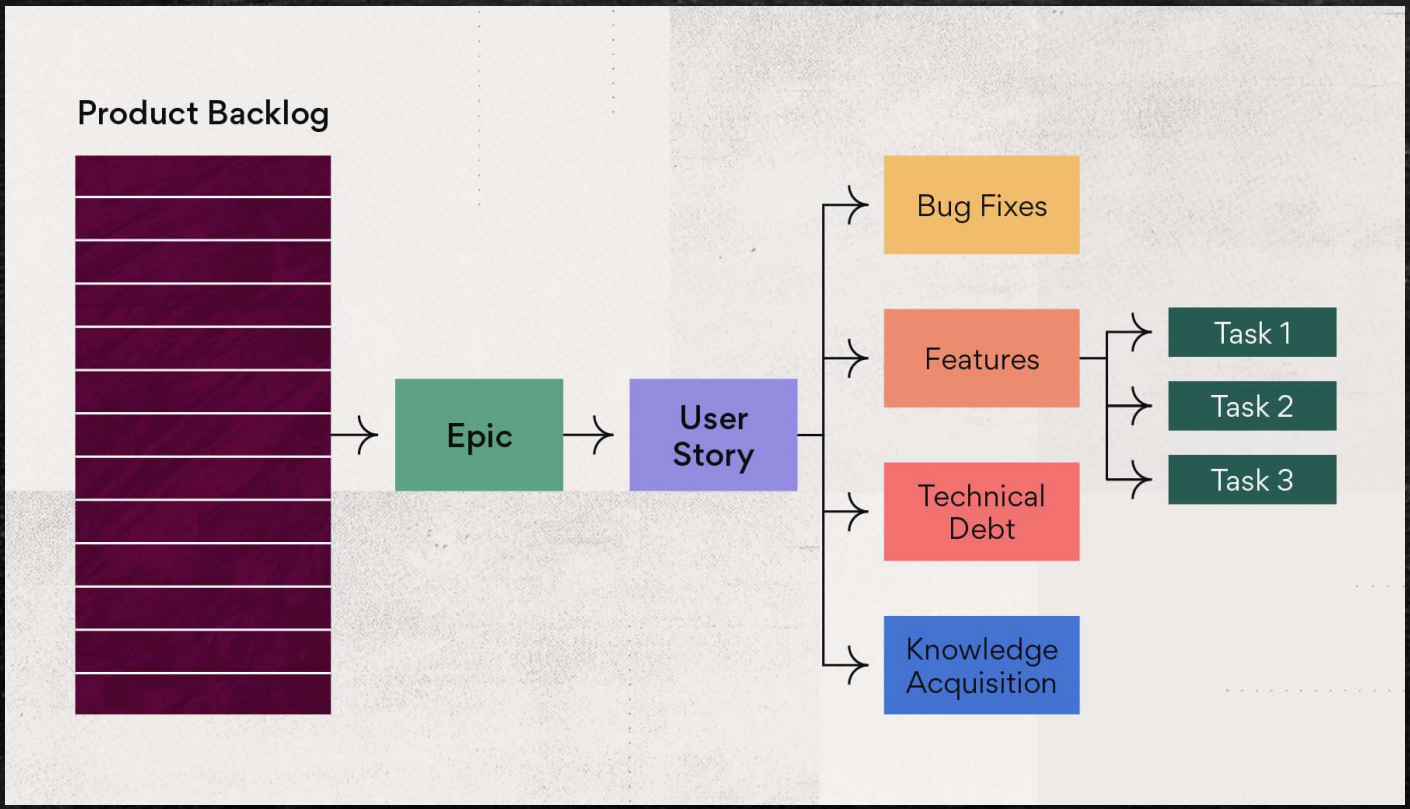
- **Gráfico Burndown**: é o gráfico utilizado pelo time scrum para **representar** diariamente o **progresso do trabalho** em desenvolvimento.
  - Ajuda a verificar se o que foi planejado está sendo realizado conforme o prazo estipulado.



Scrum

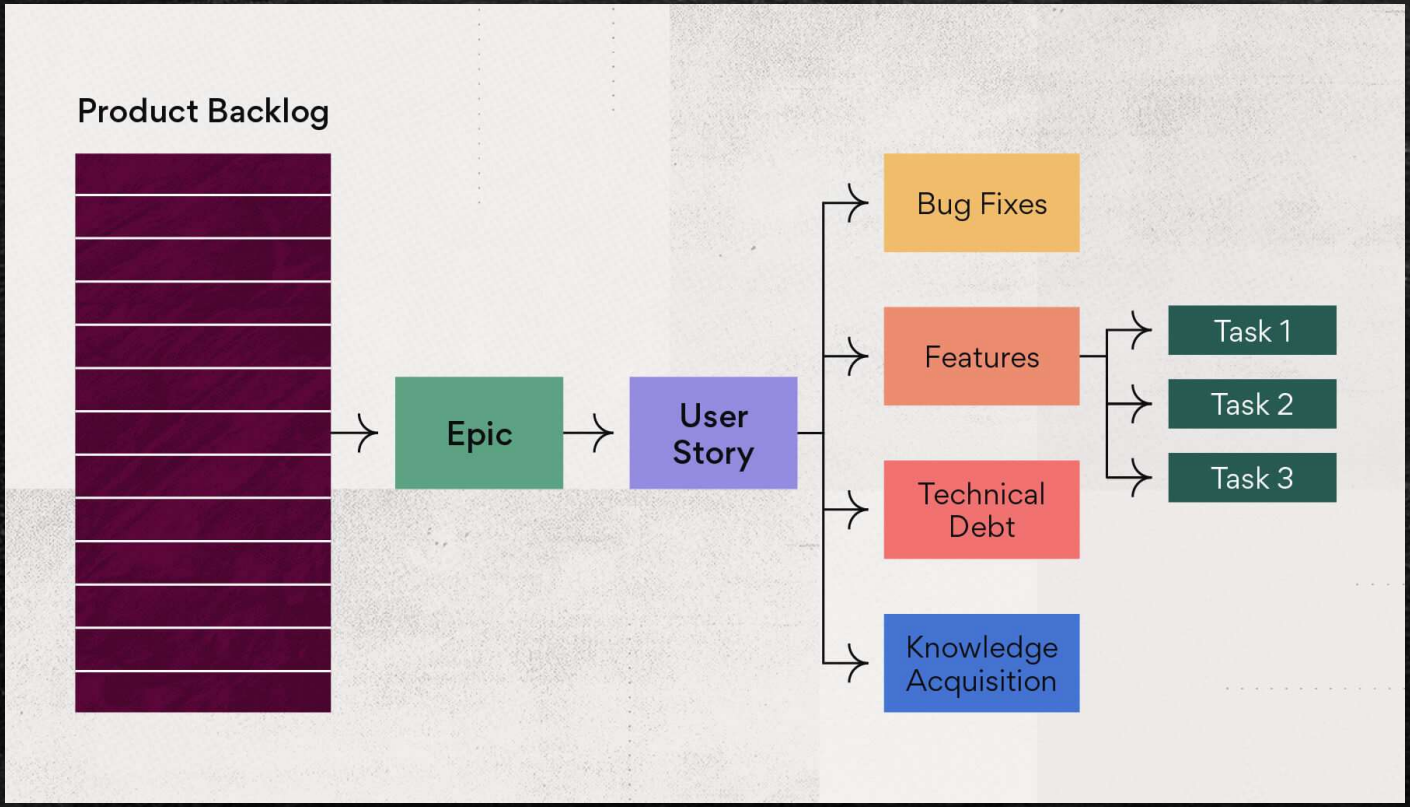
Epic: Como gerente de marketing, quero um sistema de gerenciamento de conteúdo que me permita entregar conteúdo de qualidade aos meus leitores.

User Story: Como criador de conteúdo, quero um sistema de gerenciamento de conteúdo que me permita criar conteúdo para informar os clientes sobre nossos produtos.



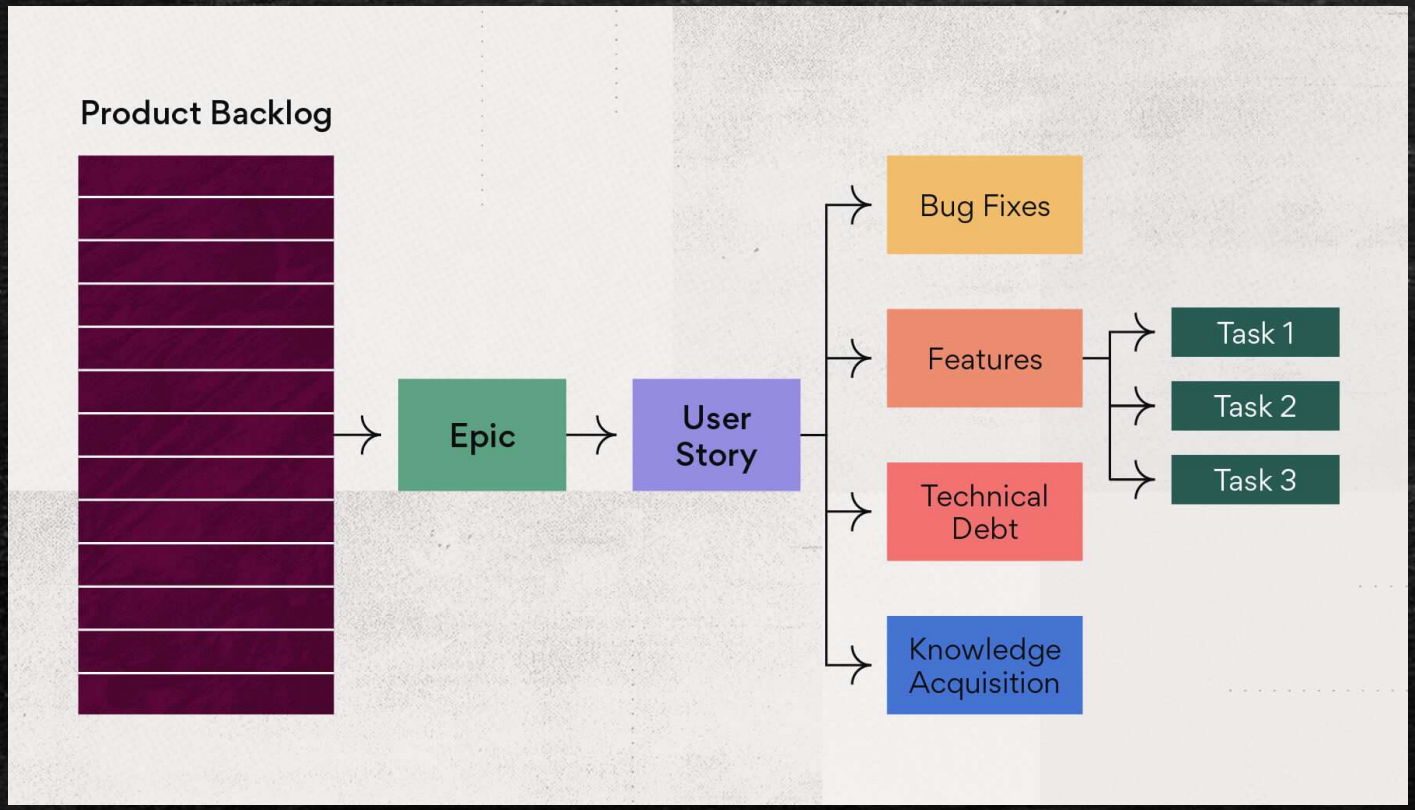
User Story (histórias de usuários): é uma **função do produto** que o usuário considera valiosa. Os recursos podem ser complexos (geralmente chamados de épicos) ou podem ser simples.

Scrum



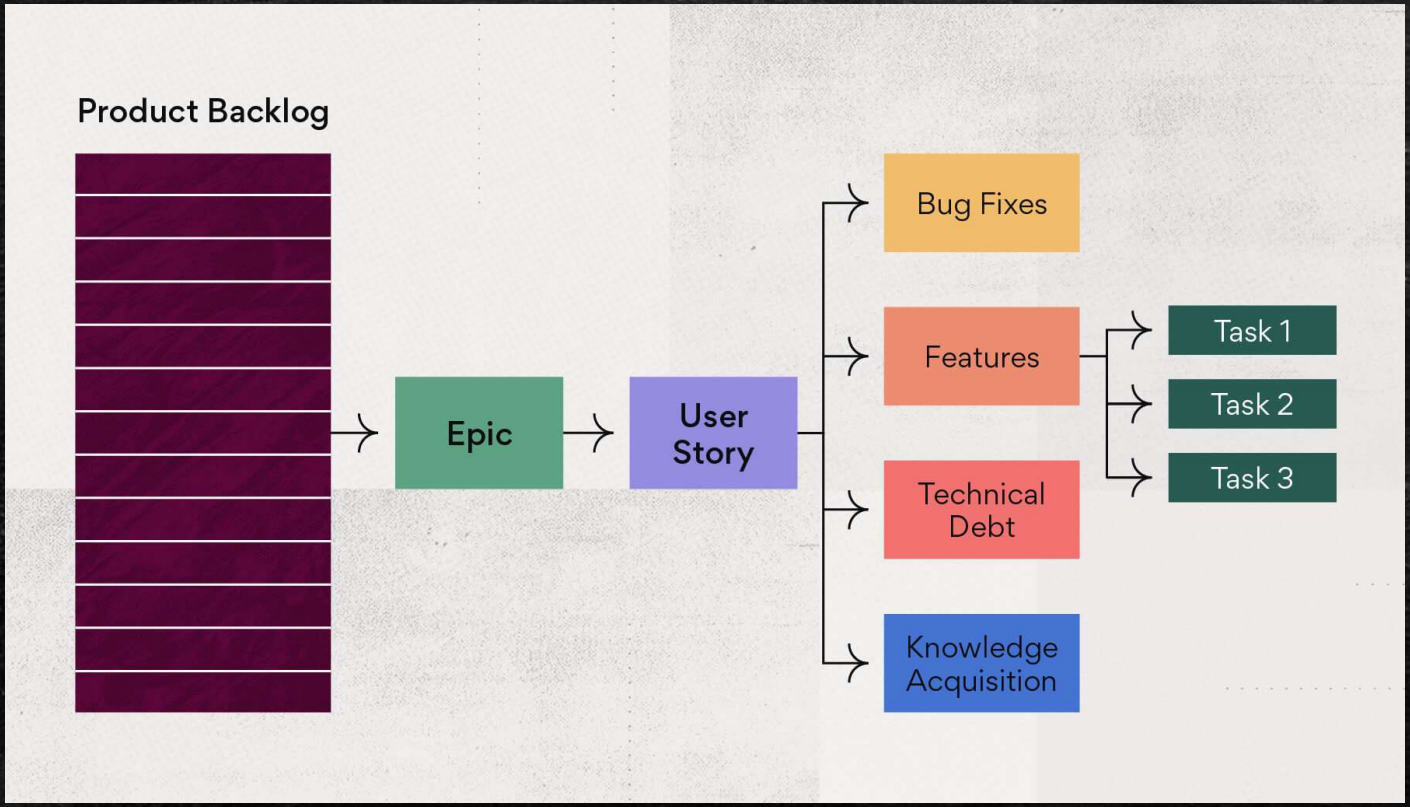
Correções de bugs: são autoexplicativas e sua equipe deve abordá-las rapidamente para manter a integridade do produto. Alguns bugs podem ser importantes o suficiente e interromper sprint atual ou podem esperar pela próxima sprint. Uma regra geral com bugs é mantê-los no topo do PB para que a equipe não esqueça.

Scrum



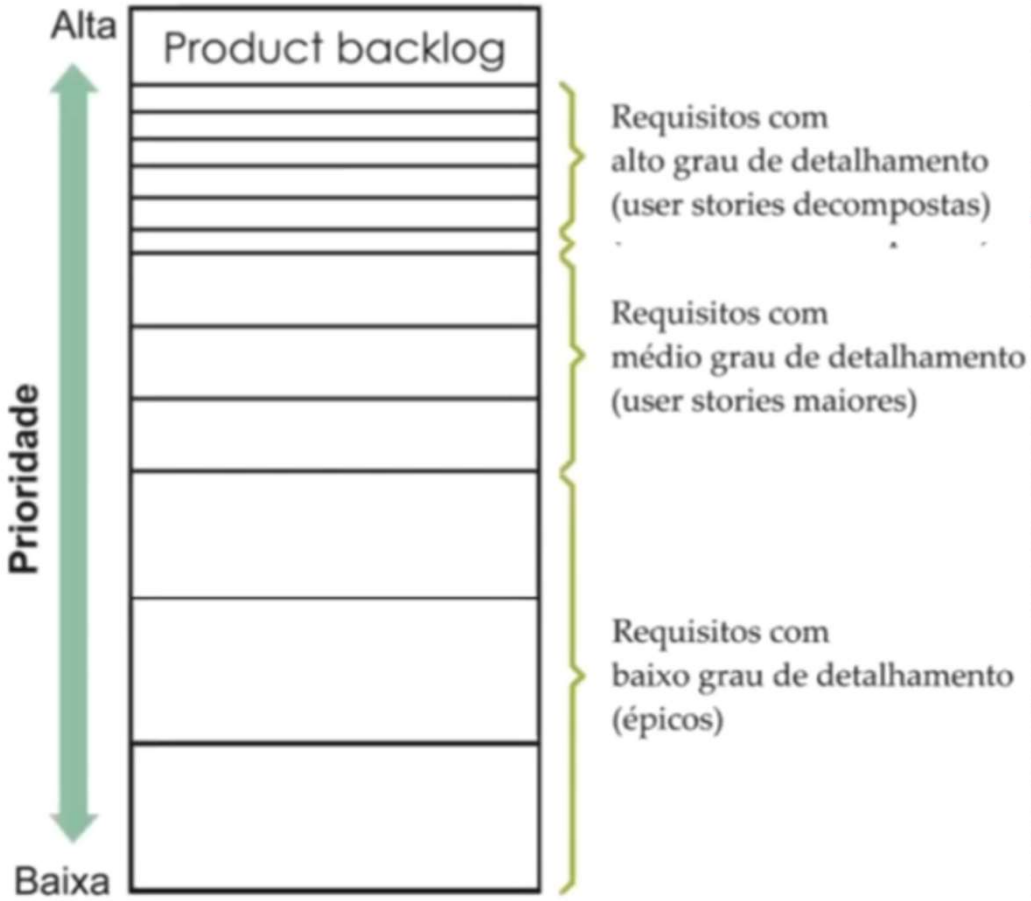
Dívidas técnicas: assim como a dívida financeira, "acumula juros" quando ignorada. Quando os desenvolvedores empurram o trabalho técnico para o fundo do PB, ele se acumula e se torna mais difícil de realizar. O gerenciamento eficaz do backlog pode evitar o acúmulo de dívida técnica.

Scrum



Aquisição de conhecimento: coleta-se informações para realizar tarefas futuras, sendo esta uma fase de pesquisa. Ao identificar um recurso que precisa de mais pesquisa, você cria uma tarefa de aquisição de conhecimento, como um protótipo, um experimento ou uma prova de conceito, para obter as informações necessárias.

Scrum



Prioridade dos requisitos do product backlog determina o grau de detalhamento

Scrum

User Story: Como criador de conteúdo, quero um sistema de gerenciamento de conteúdo que me permita criar conteúdo para informar os clientes sobre nossos produtos.

REFINANDO

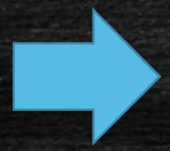
Recursos que o produto deve incluir para a User Story:

1. Faça login no sistema de gerenciamento de conteúdo
2. Criar conteúdo
3. Editar uma página de conteúdo
4. Salvar alterações
5. Atribuir conteúdo ao editor para revisão

Scrum

- Criando Histórias de Usuário:
  - Responsabilidade do PO

Eu, enquanto <Quem>,
quero <O quê> ,
para <Por que>



Exemplo:
Eu, enquanto < usuário do banco X >,
quero < fazer login na minha contra corrente >
para < verificar meu saldo >.

Scrum

- Criando Histórias de Usuário:
  - Formato Connextra (2001):

Como (papel), eu quero (algo) para (me beneficiar)

Exemplo:
Como um **gerente** do hotel, eu quero **estabelecer taxas ótimas** nos quartos do hotel para **aumentar número de clientes**.

Scrum

Funcionalidade:
Controlar o estoque dos produtos

Como **gerente de estoque,** eu quero **obter informações sobre os produtos em estoque** para **não permitir que falte um produto**.

Cenário 1: Ajustar estoque

- **Dado** que estou no limite mínimo de quantidade de um produto.
- **E** tenho uma filial com estoque do produto.
- **E** transportadora entre filial tem espaço.
- **Quando** transporte for solicitado entre filial
- **Então** enviar 50% do estoque do produto.

Cenário 2: Solicitar produto

- **Dado** que estou no limite mínimo de quantidade de um produto.
- **E** não tenho uma filial com estoque do produto.
- **Quando** fornecedor informar remessa
- **Então** solicitar mais X produtos.

Scrum

User Story

Estória do Usuário
<Nome do Projeto>

Estimativa em horas: X

< Nº > : Nome da Estória

- Breve descrição, frases curtas definindo a funcionalidade ou restrição

Prioridade da Estória

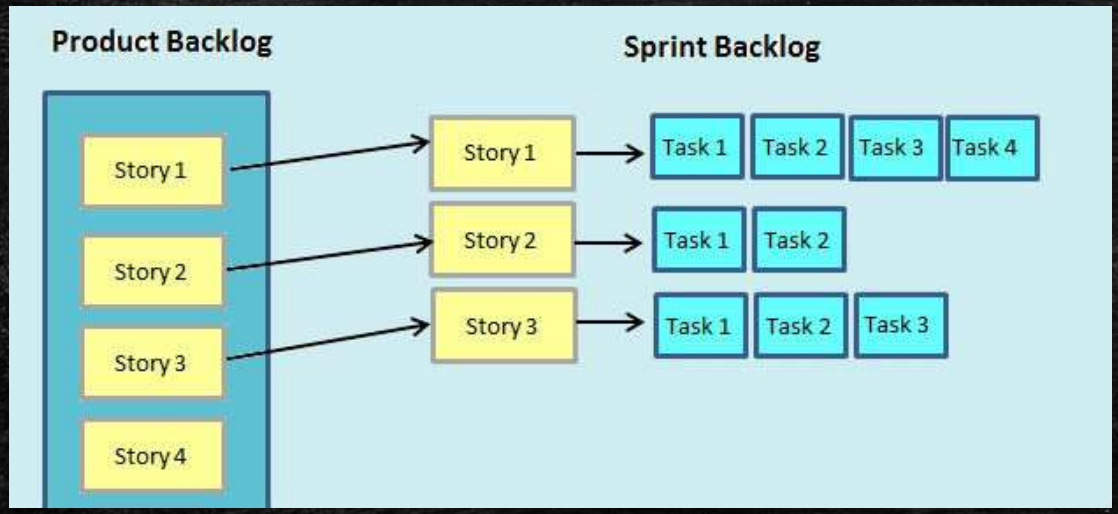
- Essencial
- Importante
- Desejável

TAREFAS:

- Lista de tarefas da Estória
- Ex: Adicionar o formulário para cadastro de usuários

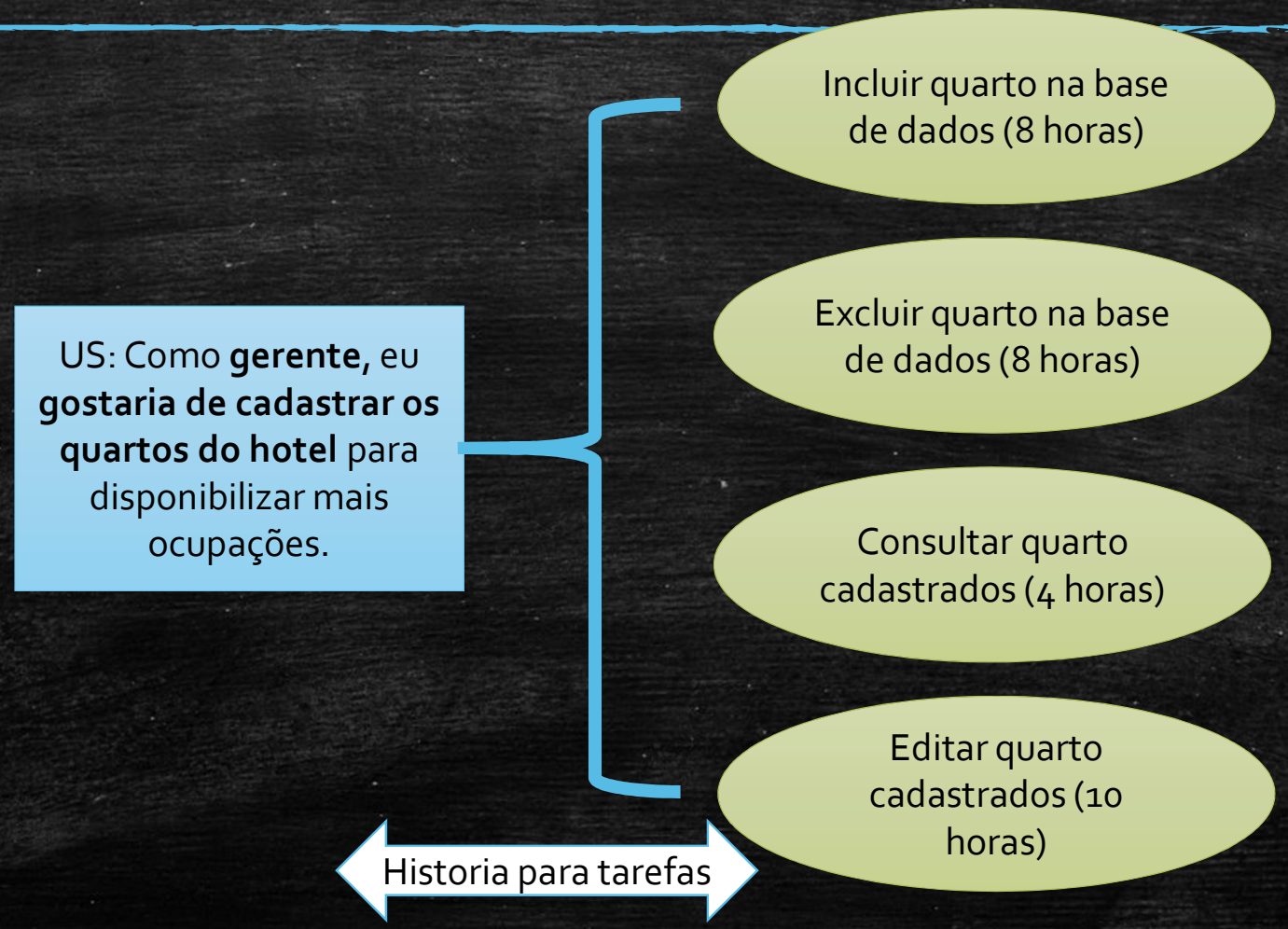
Scrum

- **Sprint Backlog:** é uma lista de tarefas que o *Scrum Team* se compromete a fazer em uma Sprint.



Os **itens do Sprint Backlog** são extraídos do Product Backlog (PB), pela equipe, com base nas prioridades definidas pelo PO e a percepção da equipe sobre o tempo que será necessário para completar as várias funcionalidades.

Scrum


US: Como **gerente,** eu **gostaria de cadastrar os quartos do hotel** para disponibilizar mais ocupações.
Incluir quarto na base de dados (8 horas)
Excluir quarto na base de dados (8 horas)
Consultar quarto cadastrados (4 horas)
Editar quarto cadastrados (10 horas)
Historia para tarefas

Scrum

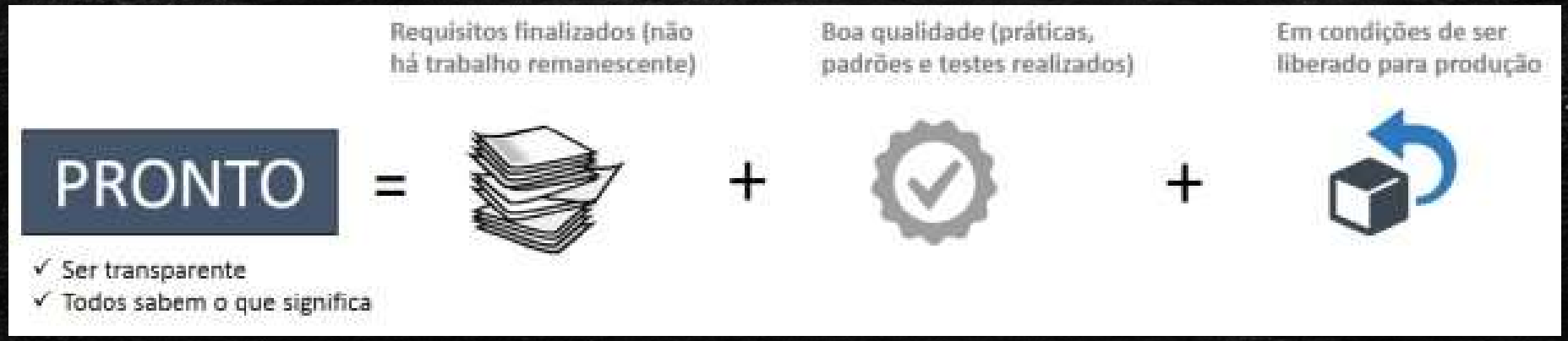
- **Planning poker** ou em português "Poker do planejamento", **é uma técnica baseada no consenso para estimar**, é um jogo e ao mesmo tempo um exercício.

Junte-se com a equipe e faça uma estimativa para a historia: Como gerente, eu gostaria de cadastrar os quartos do hotel para disponibilizar para ocupação

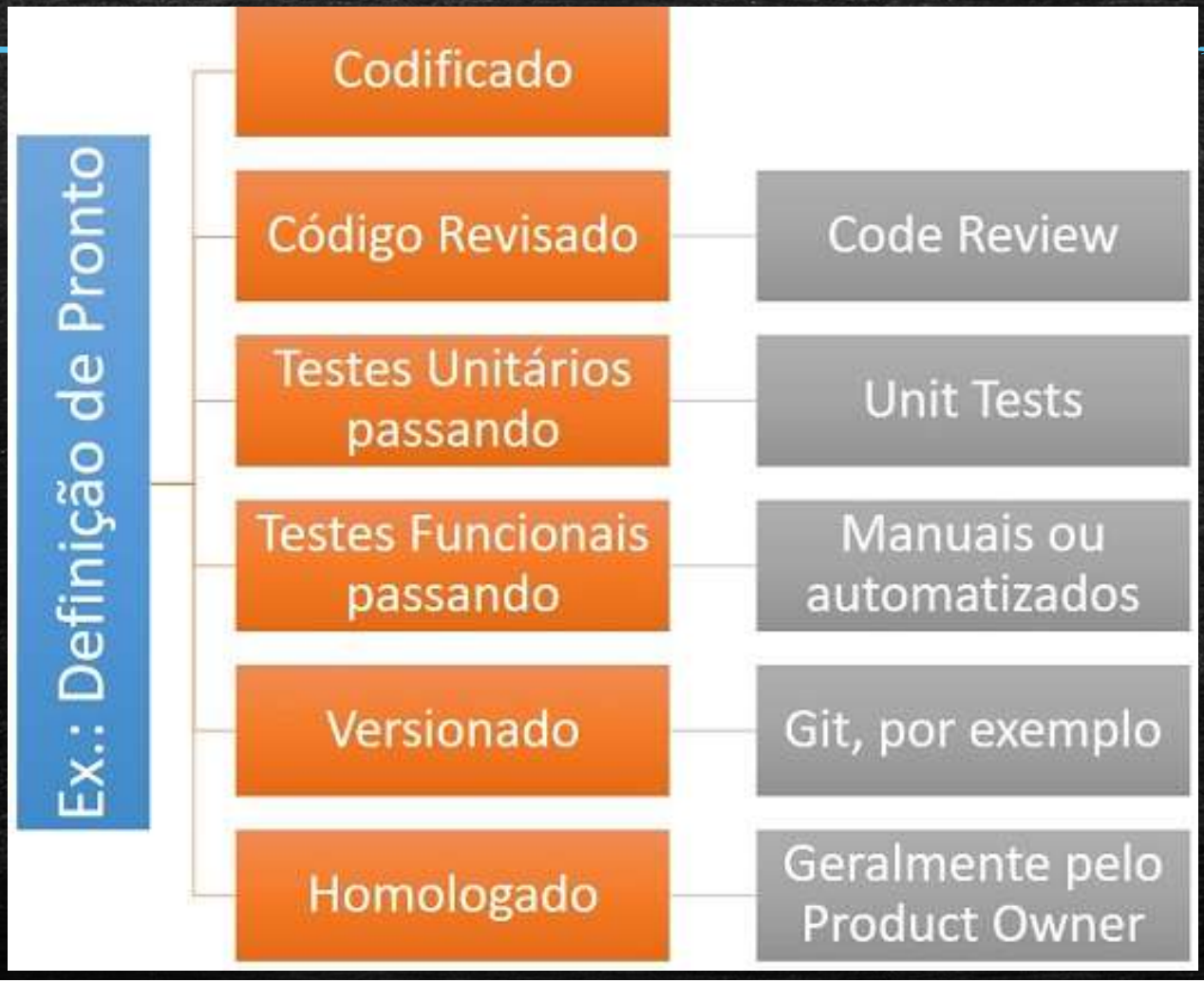
PO: perguntam "quanto tempo vai levar essa história?"
Obs: deve haver um consenso.

Scrum

- **Definição de pronto (DOD- Definition of done)** é um esforço colaborativo entre o *Scrum Team* (Devs, PO e SM)
  - O documento pode evoluir normalmente ao longo do projeto, entretanto recomenda-se que a primeira versão seja produzida durante a 1ª Sessão de Sprint Planning com seus devidos critérios de aceitação.



Scrum


Ex.: Definição de Pronto
Codificado
Código Revisado
Code Review
Testes Unitários passando
Unit Tests
Testes Funcionais passando
Manuais ou automatizados
Versionado
Git, por exemplo
Homologado
Geralmente pelo Product Owner

Scrum

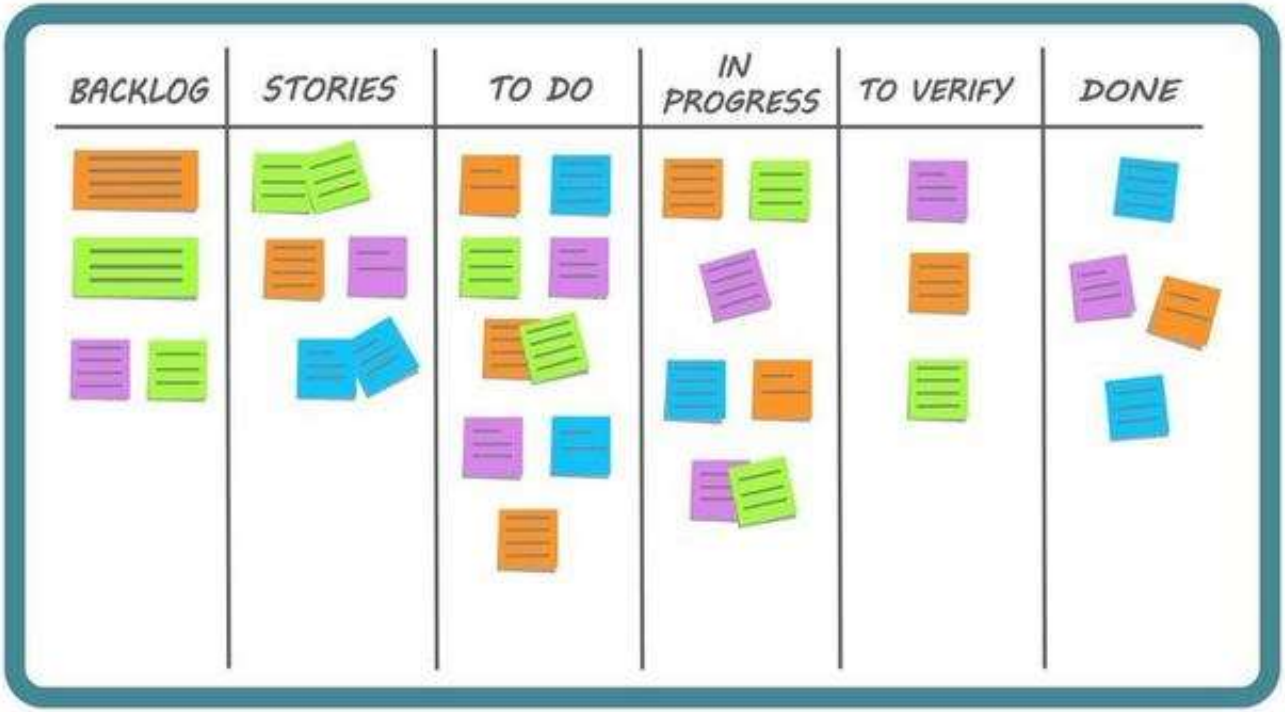
- **Kanban**: é um **quadro** de sinalização que **controla os fluxos** de produção ou transportes em uma indústria.
  - Foi criado na década de 1960 pela Toyota.
  - Já na **Engenharia de Software**, Kanban é uma **estratégia para otimizar o fluxo de valor** para partes interessadas através de um processo que utiliza um sistema visual.
  - Foi **adaptado** por David J. Anderson (fundador da Kanban University) como técnica para ser utilizada no **desenvolvimento de software**.

Scrum

- Kanban

Scrum

- Kanban



Scrum

Historias do Usuário	A fazer	Em progresso	Testando	Feito
User story 1	1. Faça login no SGC 2. Criar conteúdo 3. Editar uma página de conteúdo ⋮			
User story 2	1. Criar aba para Admin 2. Editar usuários 3. Excluir usuários ⋮			
...				

Scrum

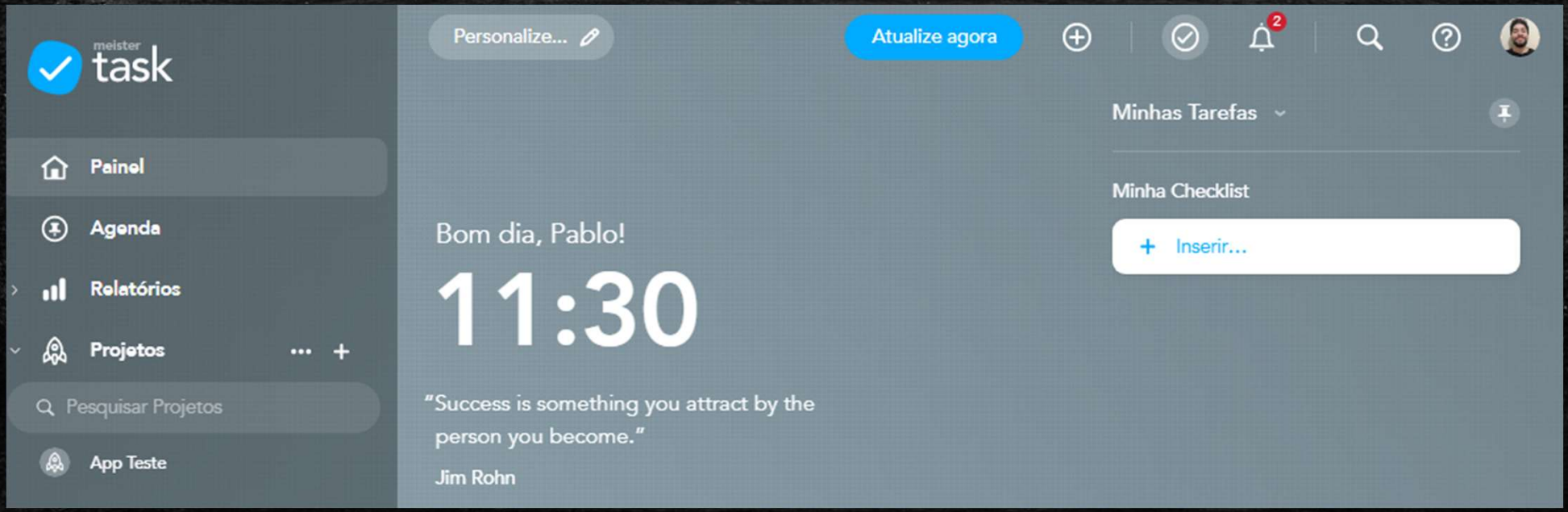
Historias do Usuário	A fazer	Em progresso	Testando	Feito
User story 1	2. Criar conteúdo 3. Editar uma página de conteúdo ⋮	1. Faça login no SGC		
User story 2	2. Editar usuários 3. Excluir usuários ⋮	1. Criar aba para Admin		
...				

Questões práticas

- Dificuldade do Cliente de fato se envolver com o projeto.
- Membros da equipe podem não ter personalidade adequada.
- Priorizar as mudanças pode ser extremamente difícil.
- Manter a simplicidade exige trabalho extra.
- Organizações tem dificuldades de mudar a forma como seus processos já são definidos e utilizados.

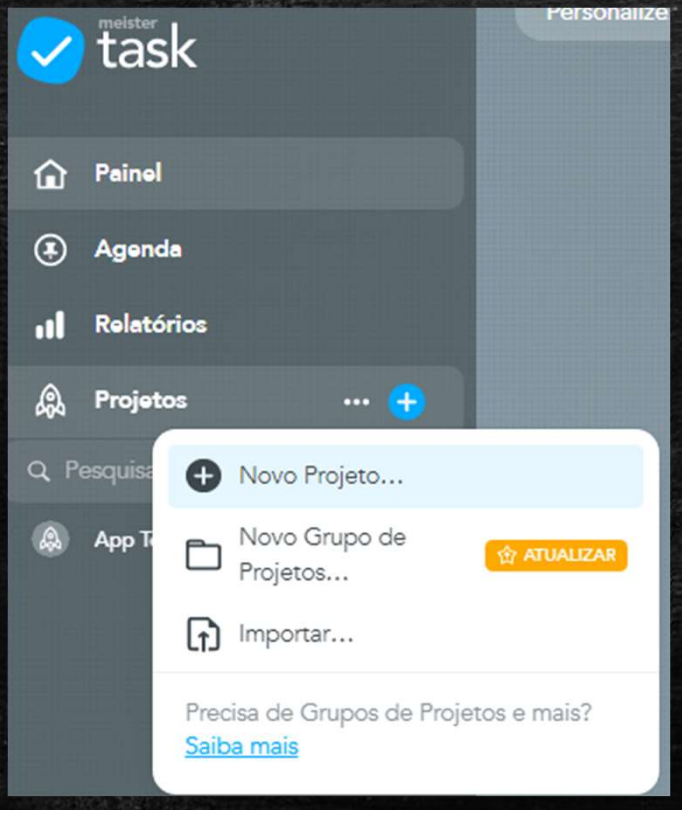
Exemplos práticas

- Abra o meistertask.com/pt
  - Crie uma conta e acesse a página do *dashboard*



Exemplos práticas

- Crie um novo projeto com o nome “Teste”



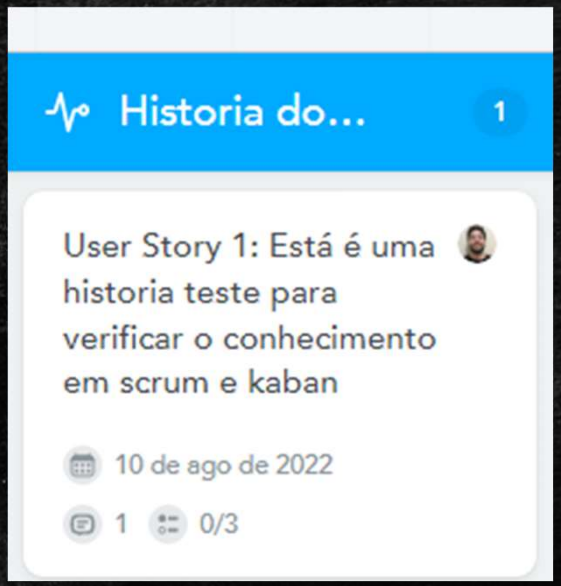
Compartilhe com sua equipe e adicione o professor (pablo.vargas@unir.br)

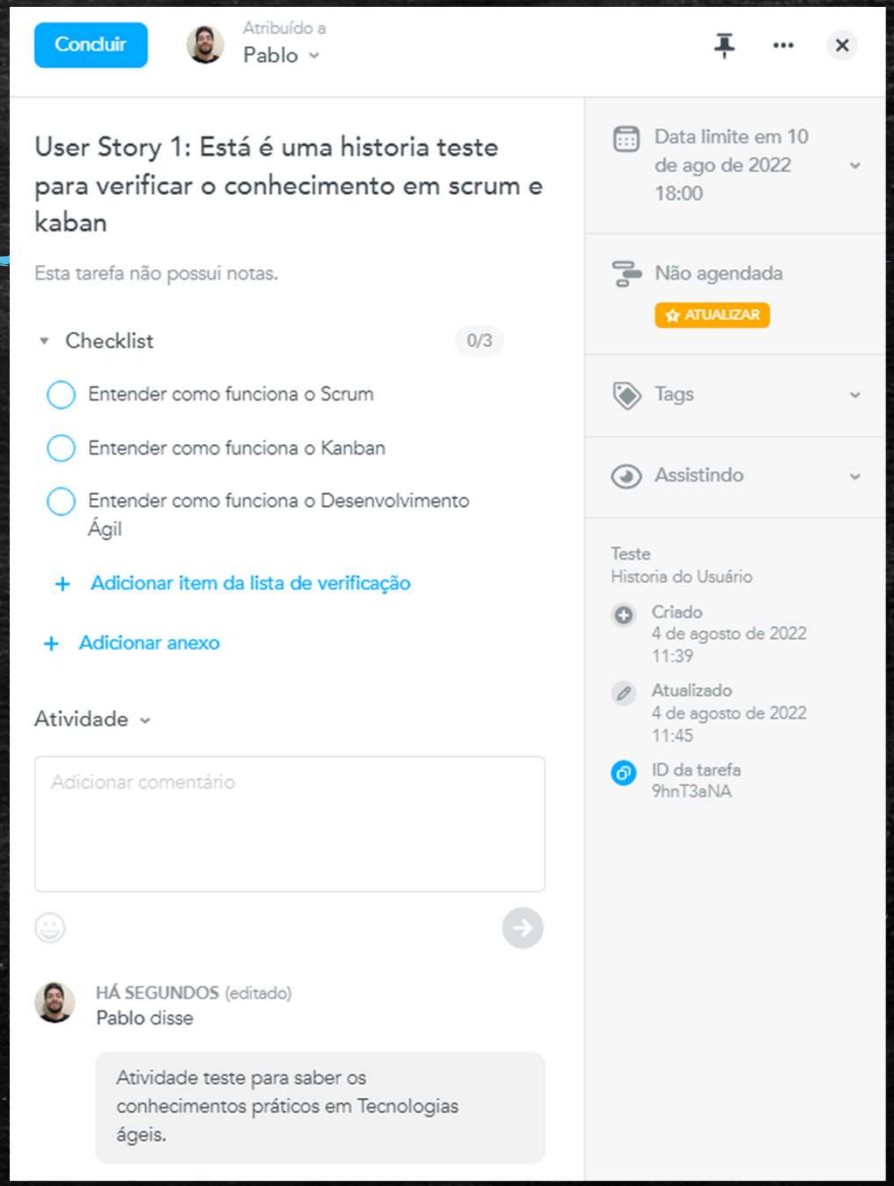
Exemplos práticas

- Crie e edite as seções de acordo com imagem abaixo

Exemplos práticas

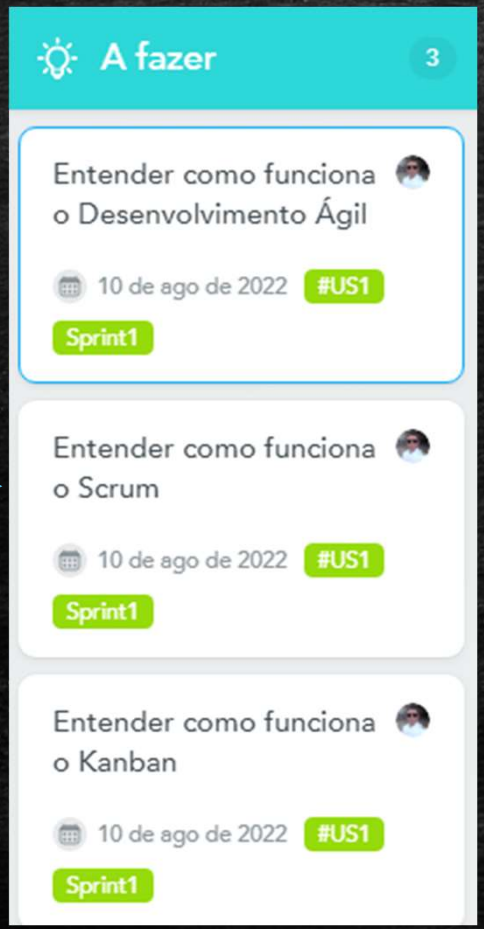
Crie uma tarefa na seção de Historia do Usuário

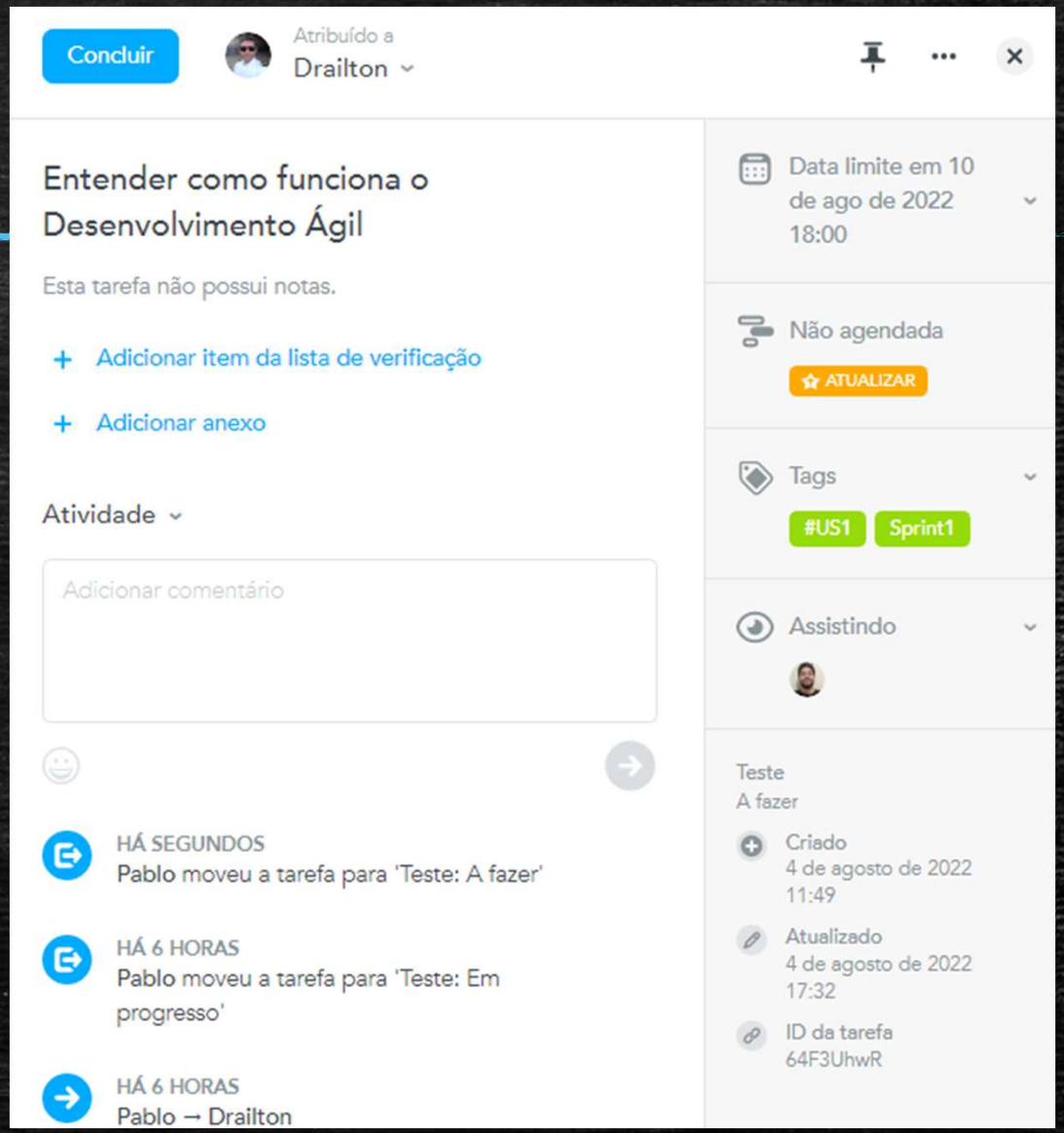




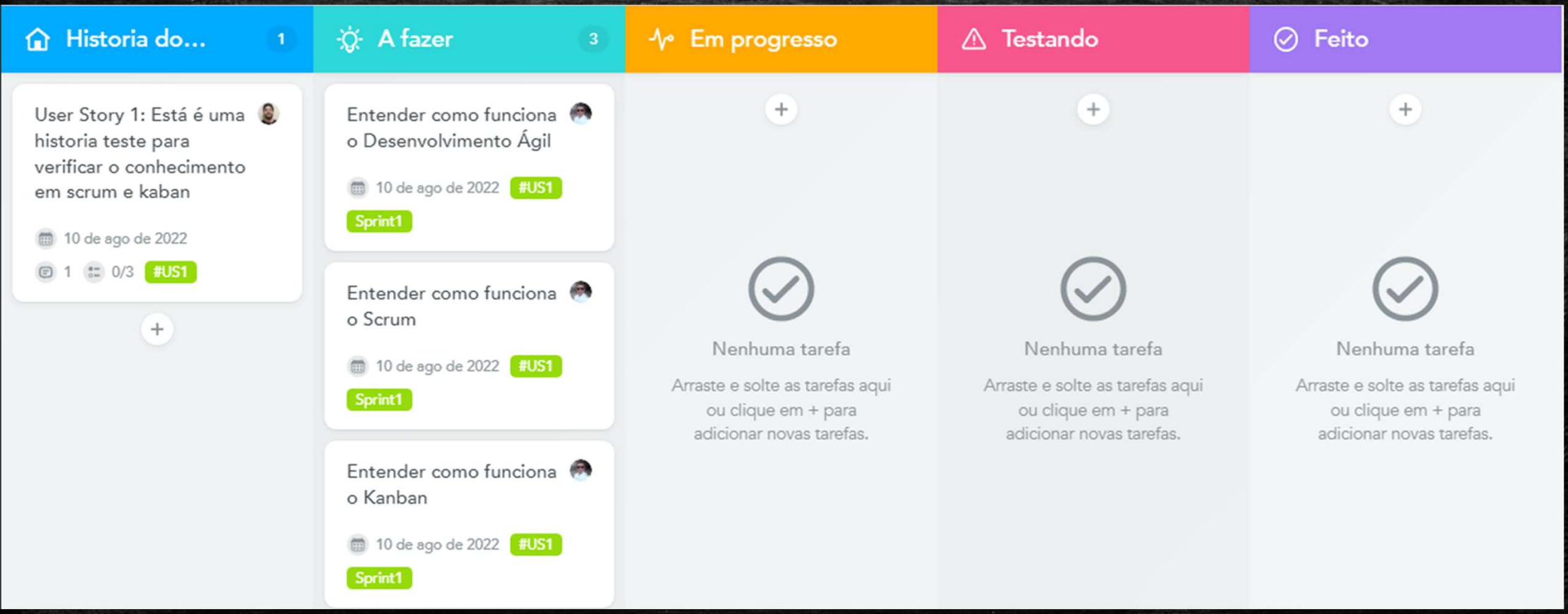
Exemplos práticas

Crie 3 tarefas na seção "A fazer"





Exemplos práticas



Exemplos práticas

- Mova uma das tarefas para “Em progresso”

Exercícios

Estipule uma data limite para as tarefas, bem como, a quem será atribuída a tarefa e uma tag.

- Crie um projeto no Meister Task para o quadro kanban abaixo com o nome Exercício 1

Historias do Usuário	A fazer	Em progresso	Testando	Feito
User story 1	1. Faça login no SGC 2. Criar conteúdo 3. Editar uma página de conteúdo ⋮			
User story 2	1. Criar aba para Admin 2. Editar usuários 3. Excluir usuários ⋮			

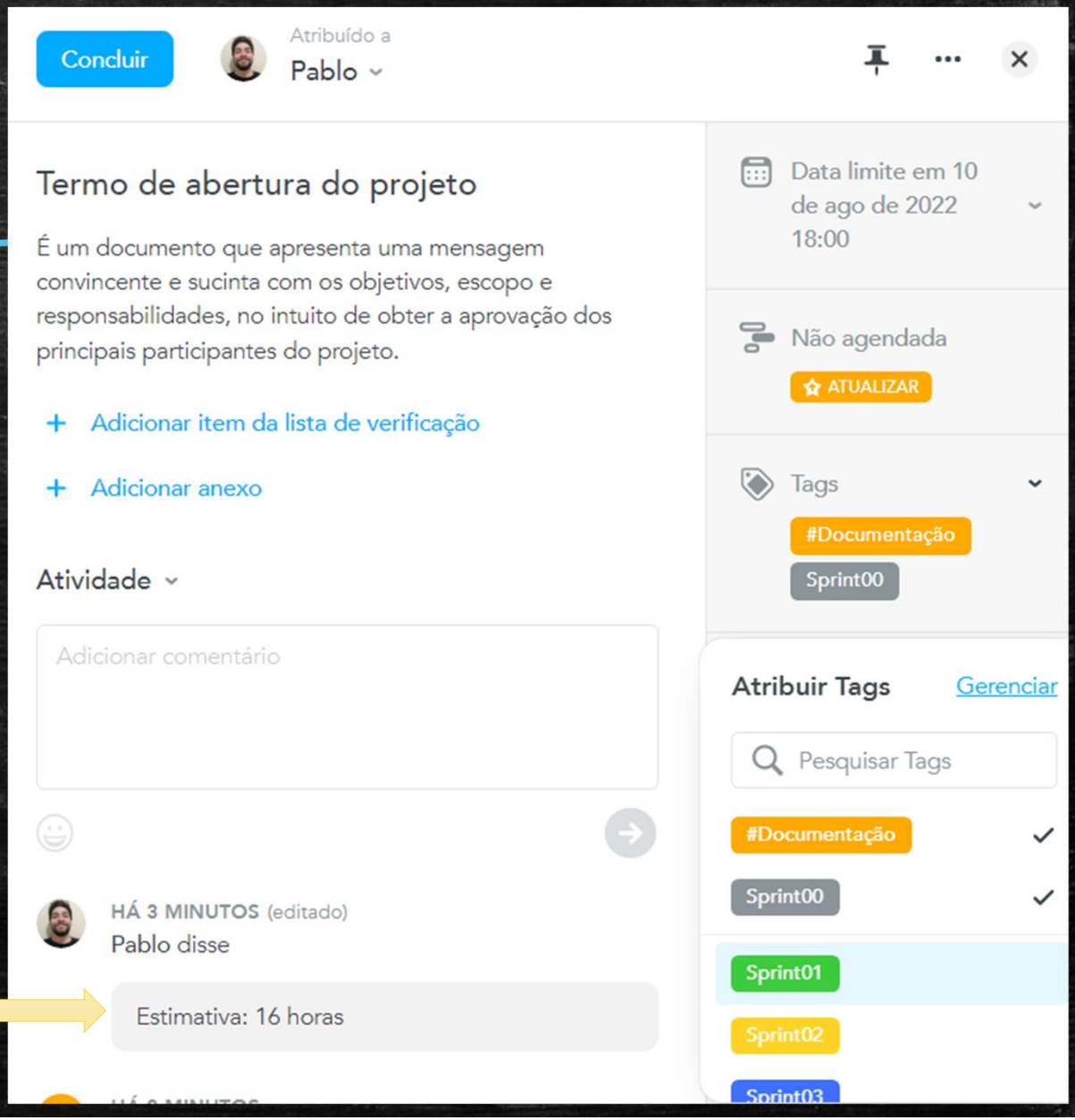
Mais exemplos

Crie um novo projeto com o nome "Exemplo 1" adicione sua equipe no projeto

Mais exemplos

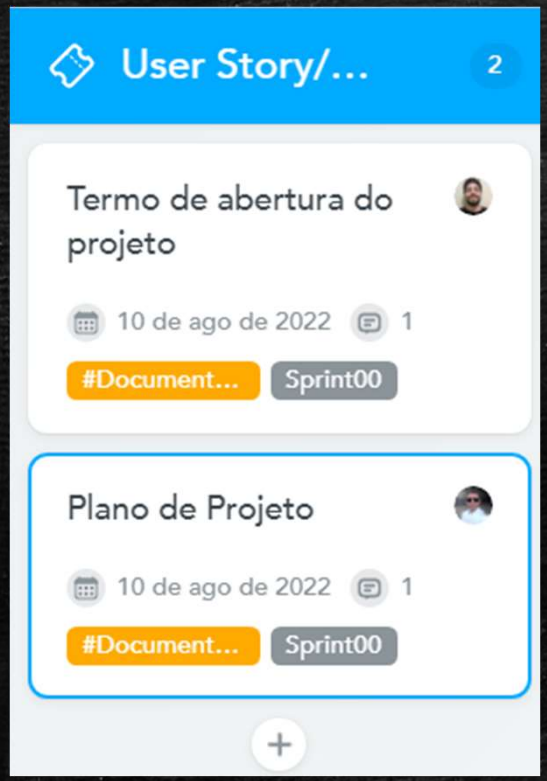
Crie as seções de acordo com a imagem abaixo.

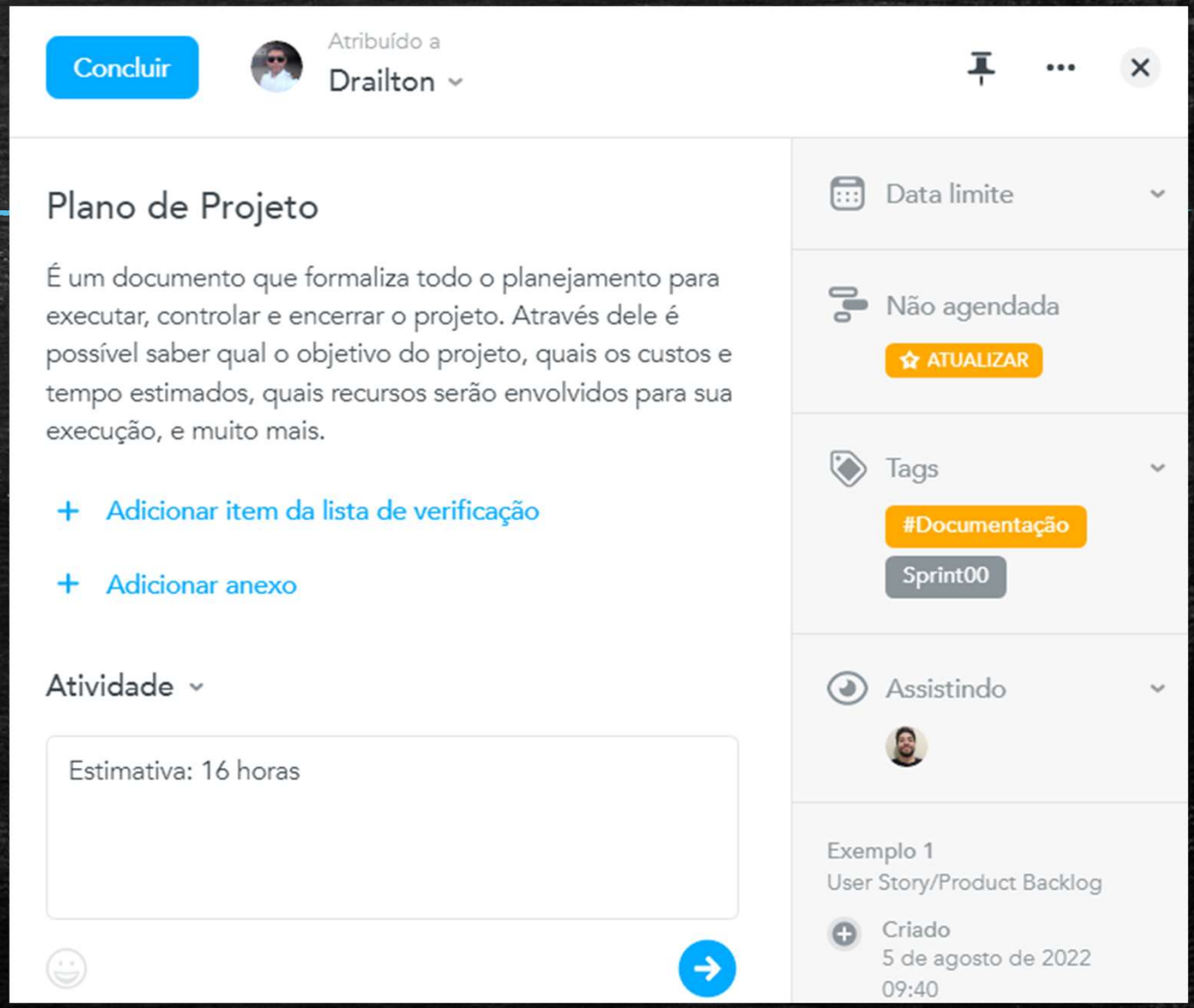
Mais exemplos



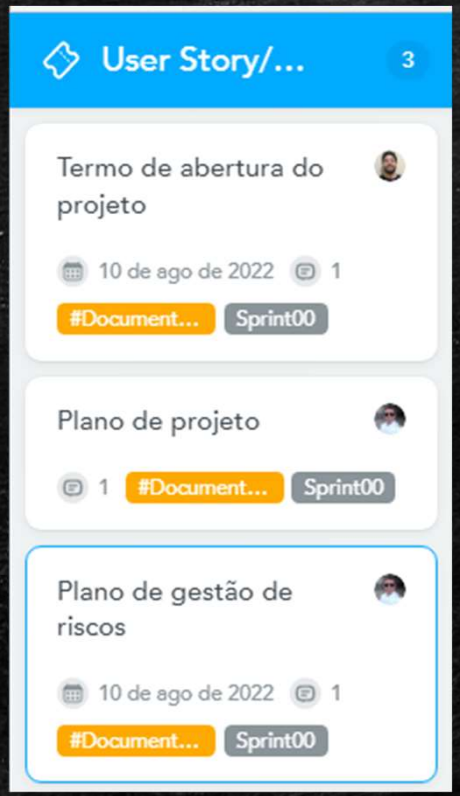
Não esqueça de adicionar, como comentário, estimativas em horas para cada tarefa criada.

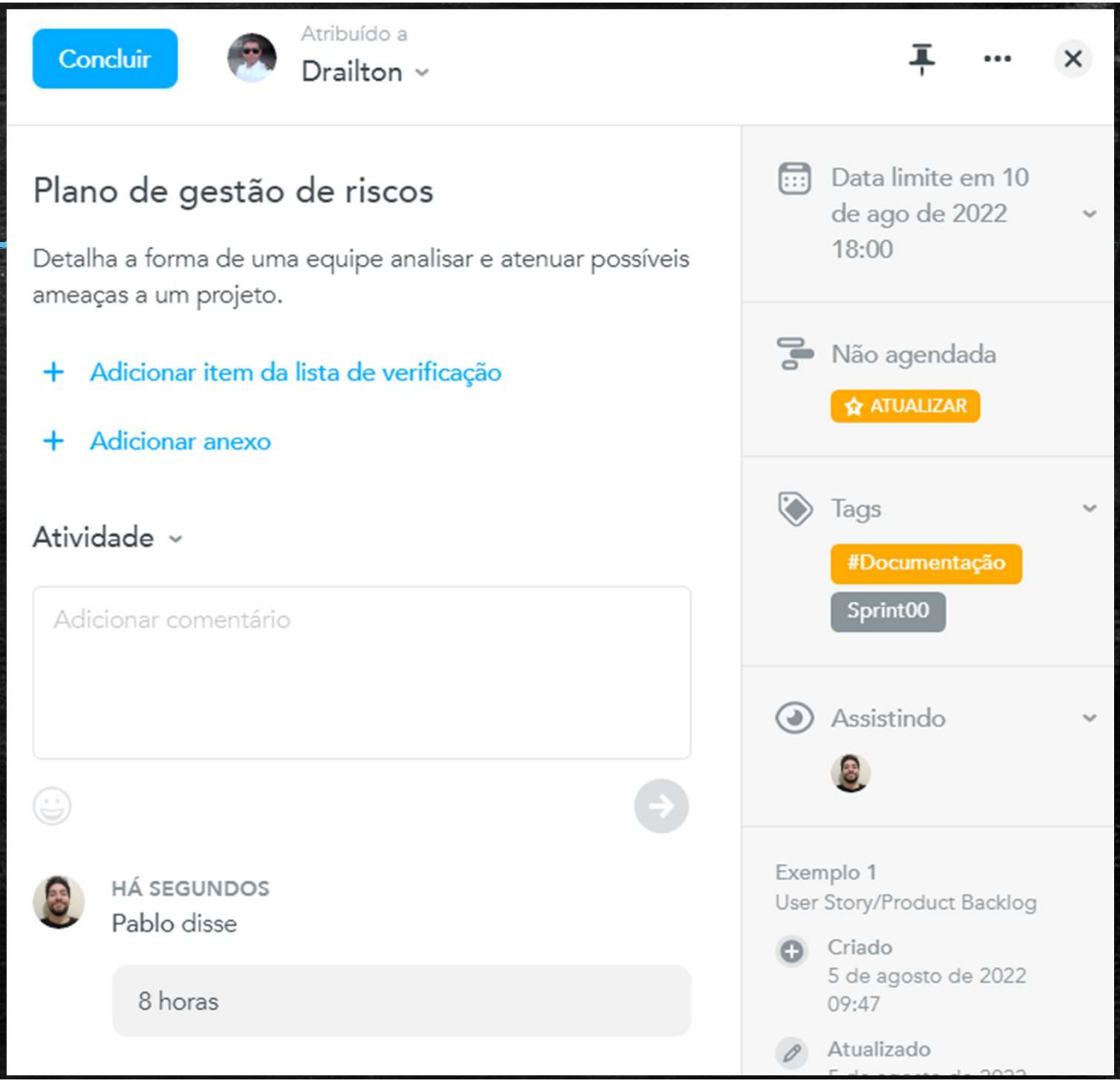
Mais exemplos



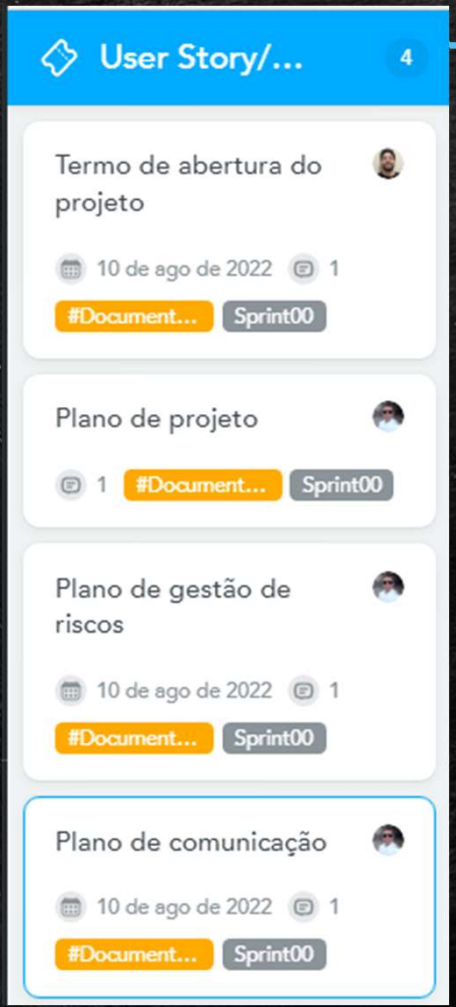


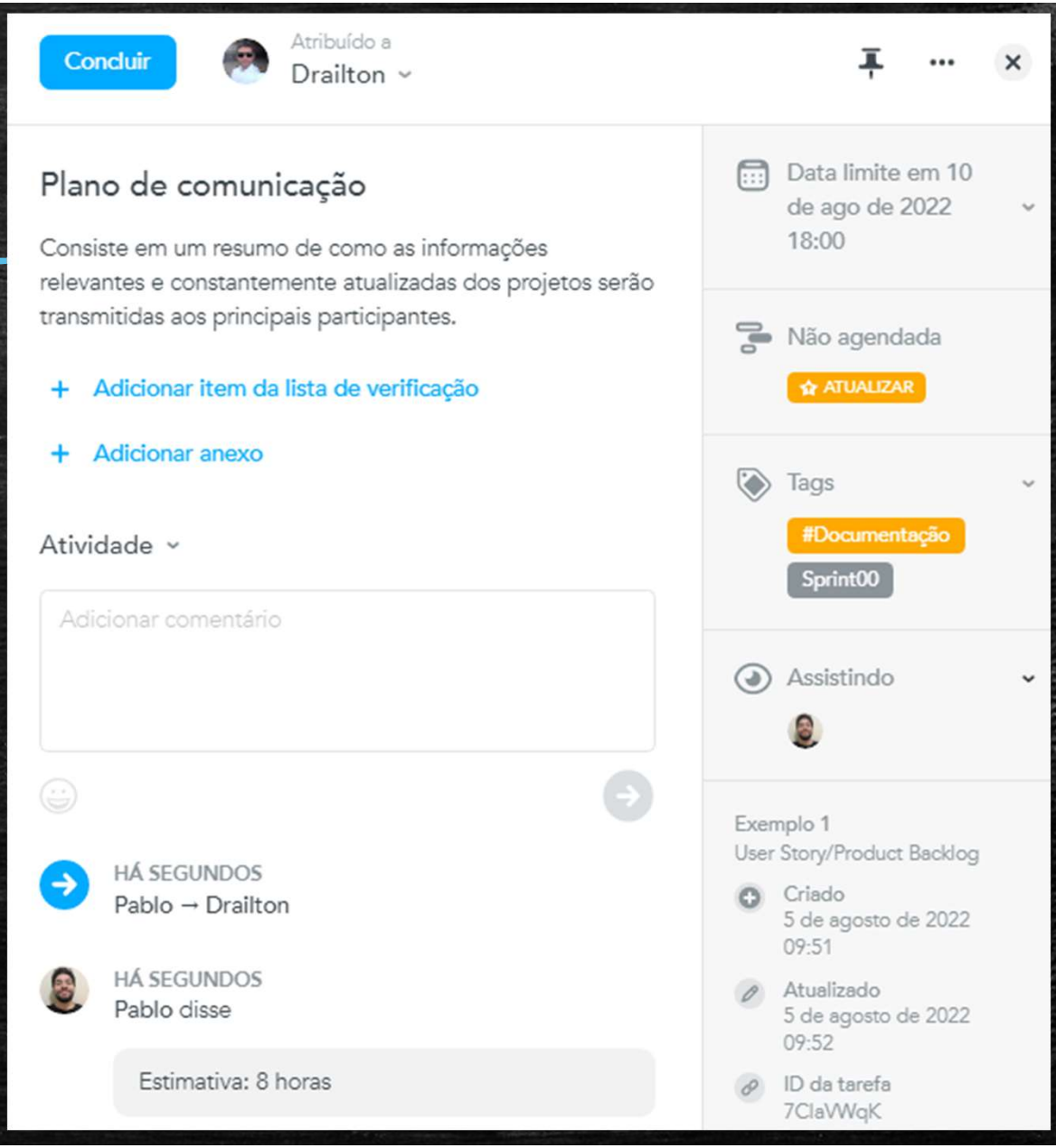
Mais exemplos



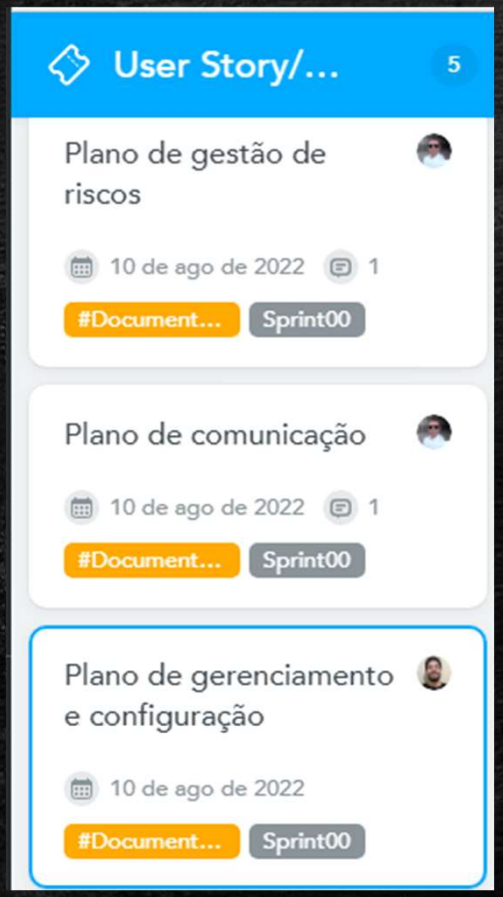


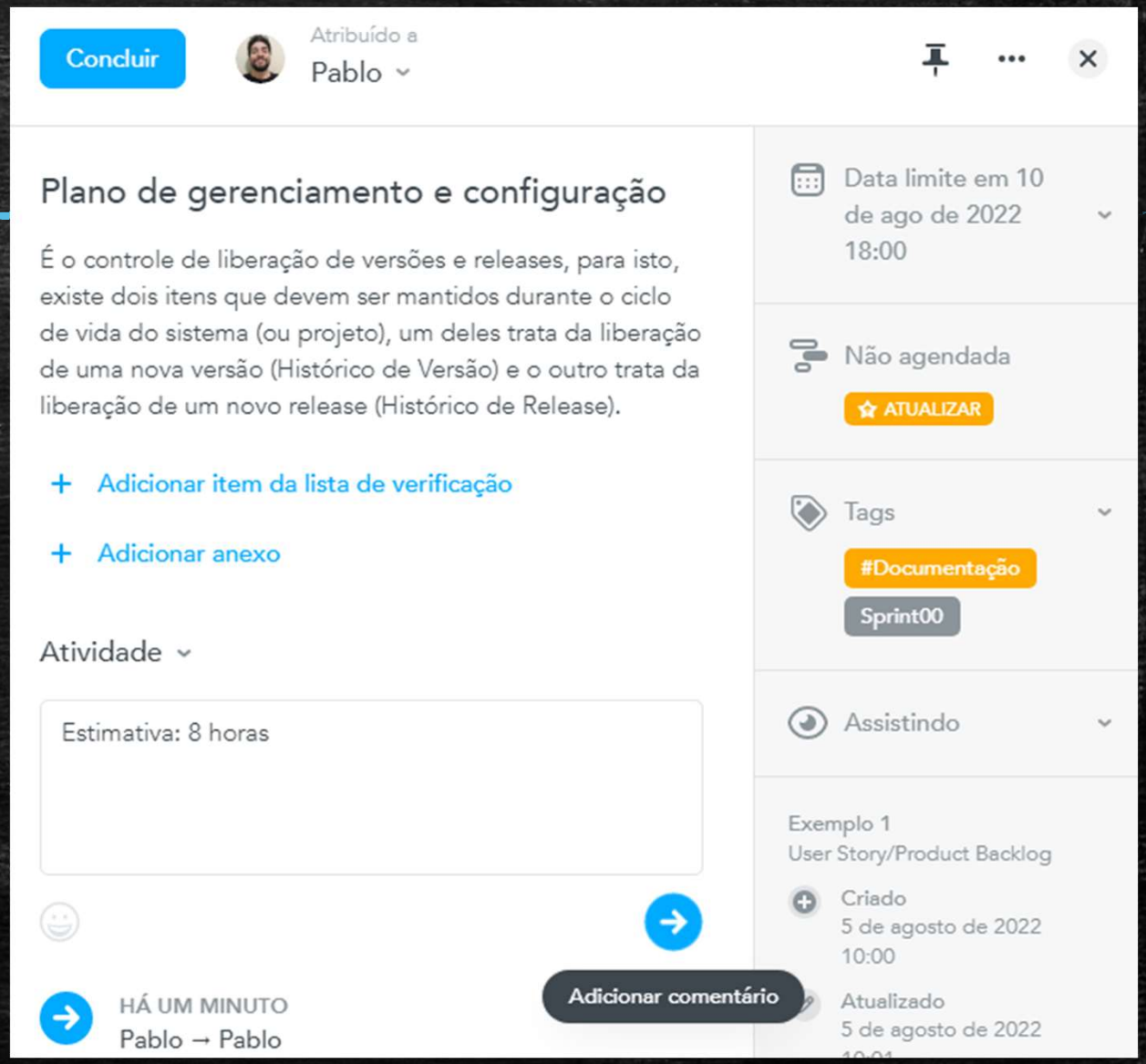
Mais exemplos



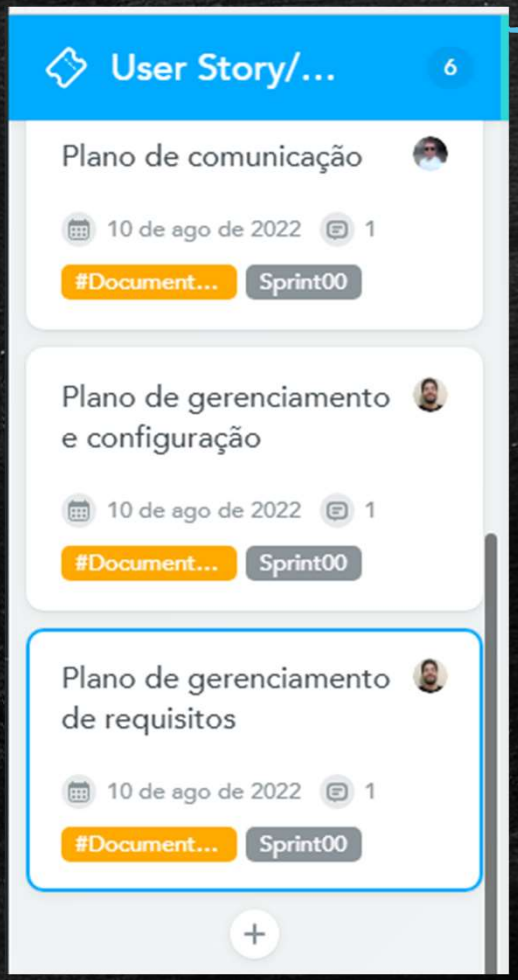


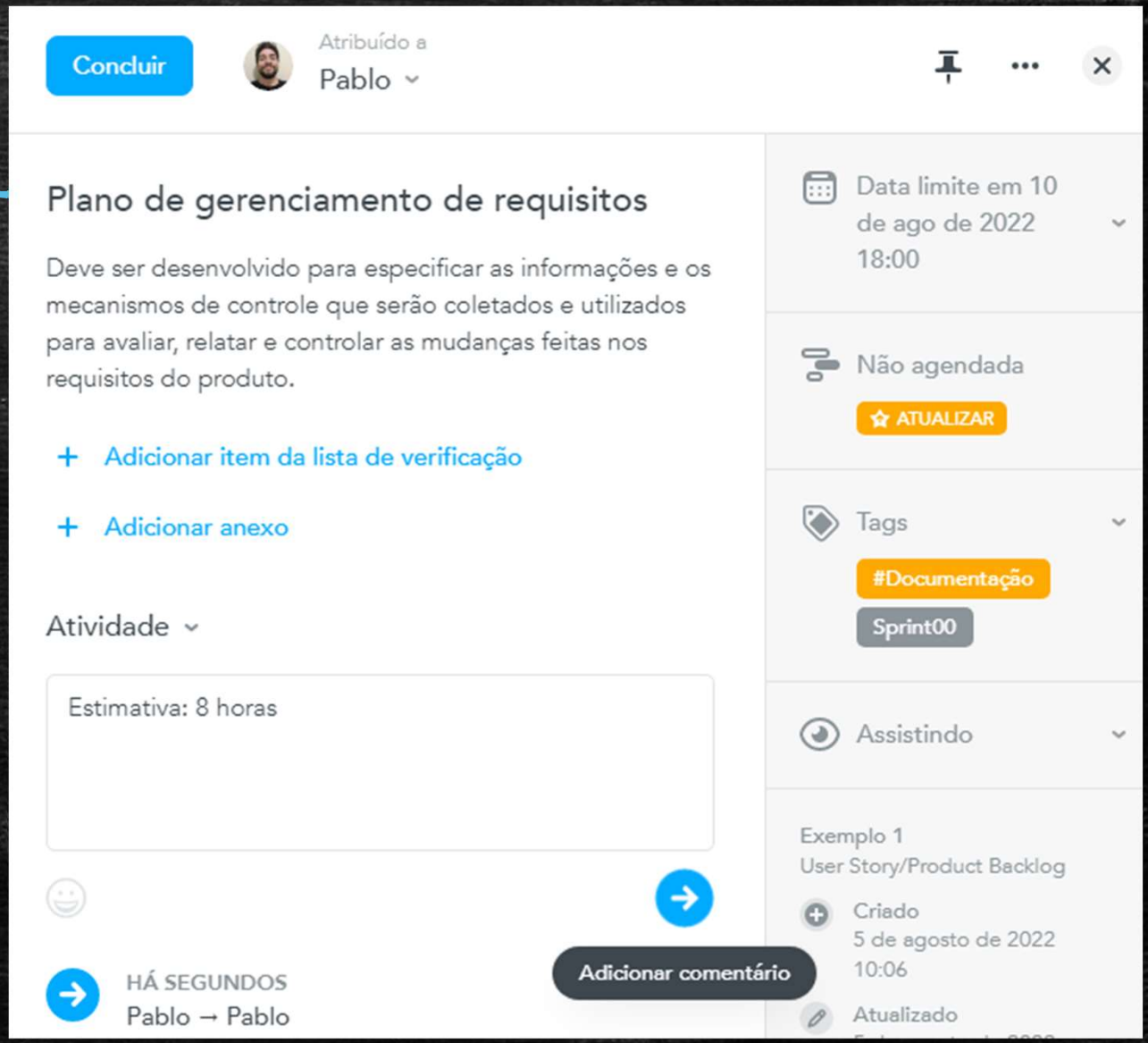
Mais exemplos





Mais exemplos

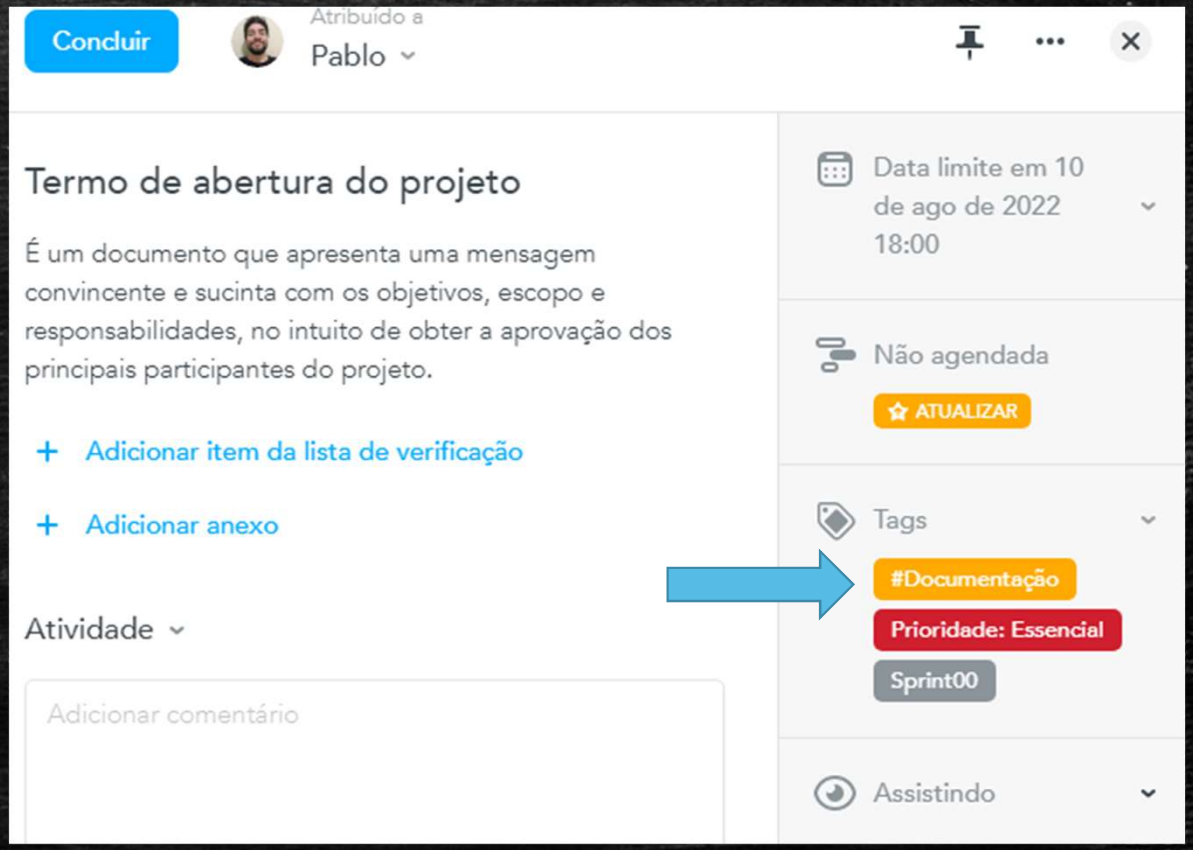




Mais exemplos

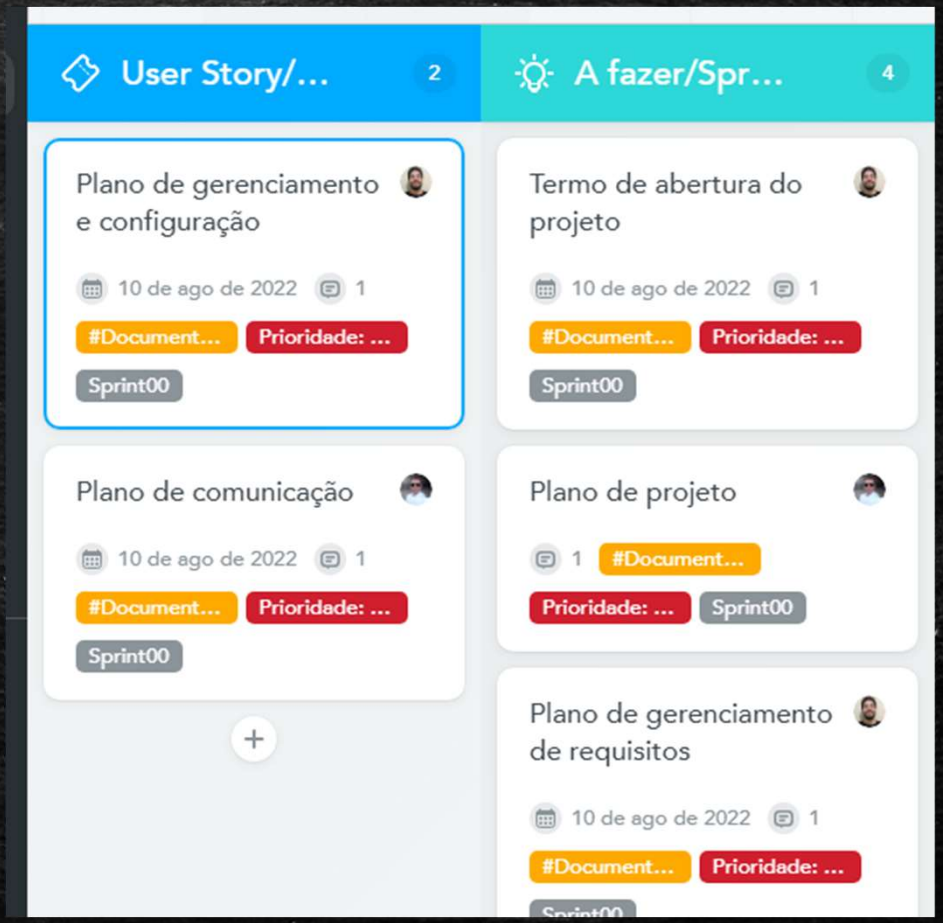
Exercício: Adicione tags de prioridade (essencial, importante e desejável) e mova os itens (essencial) do PB para seção “A fazer”.

Ex.

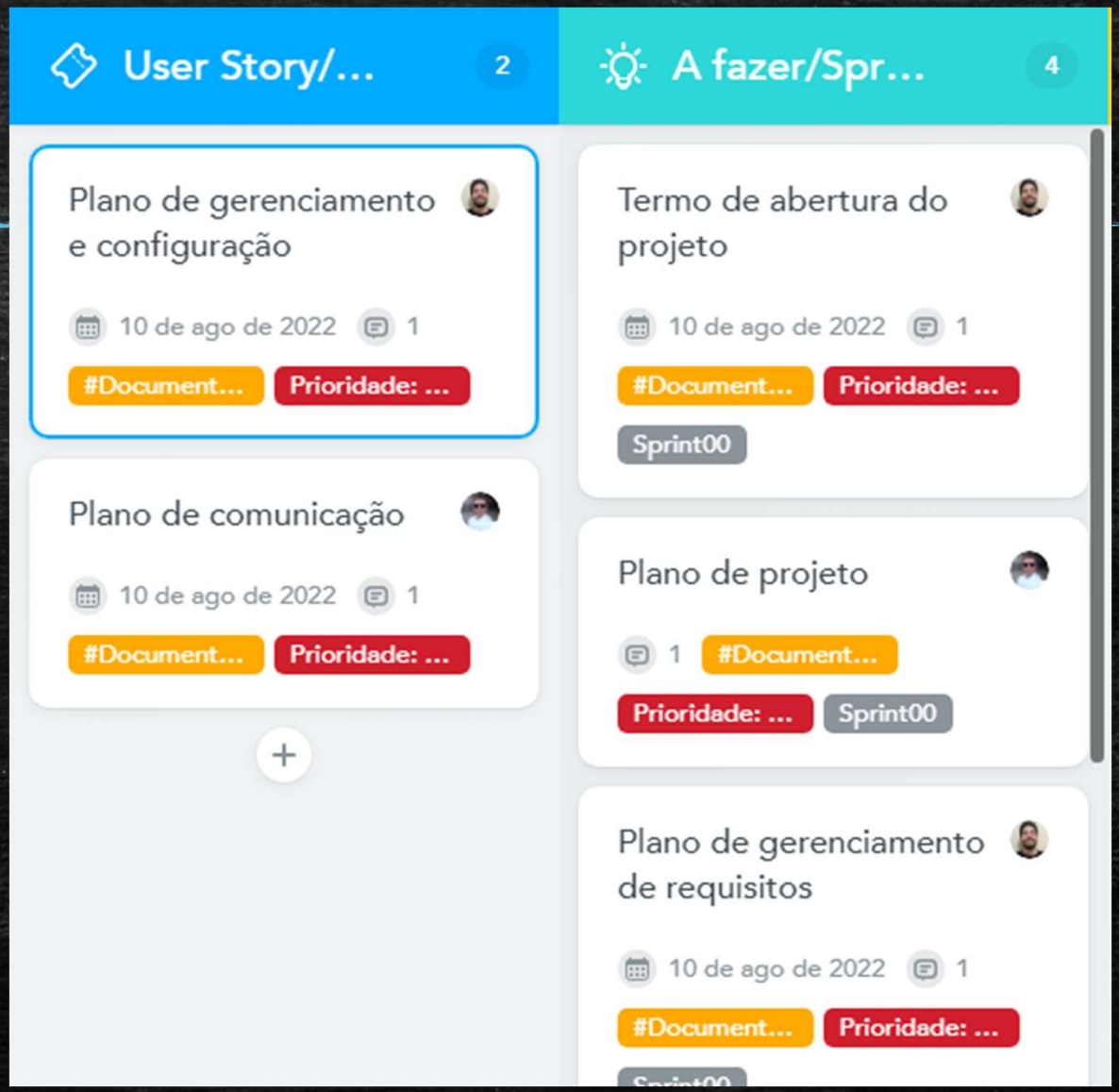


Mais exemplos

Veja que existem itens do PB que não foram para Sprint mesmo estando com tags "Sprint 00". Ou seja, você deve renomear para a próxima Sprint ou retirar a tag.

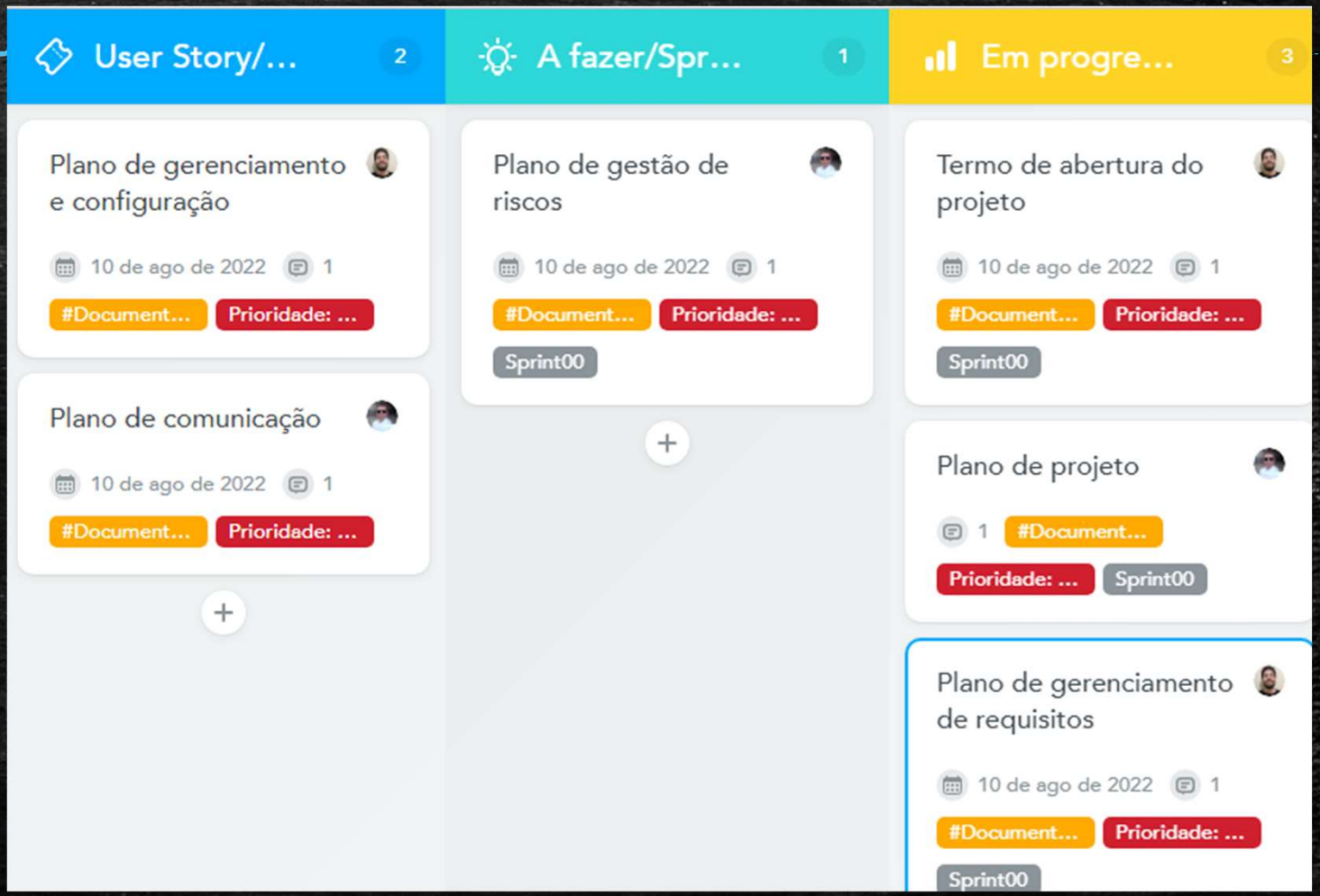


Mais exemplos



Mais exemplos

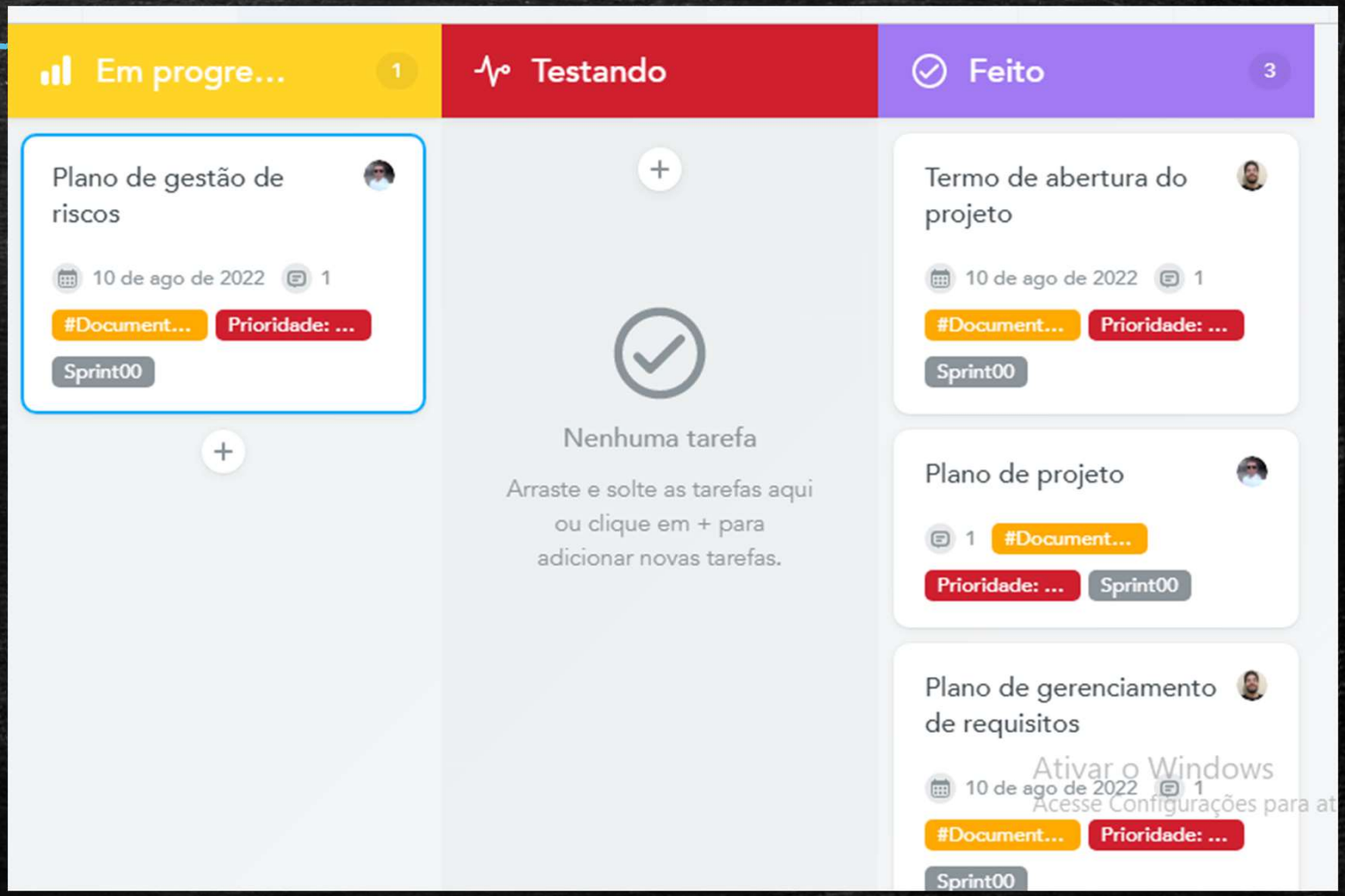
Simulando: mova algumas tarefas para "Em progresso"



Mais exemplos

Simulando: mova tarefas de "Em progresso" para "Feito"

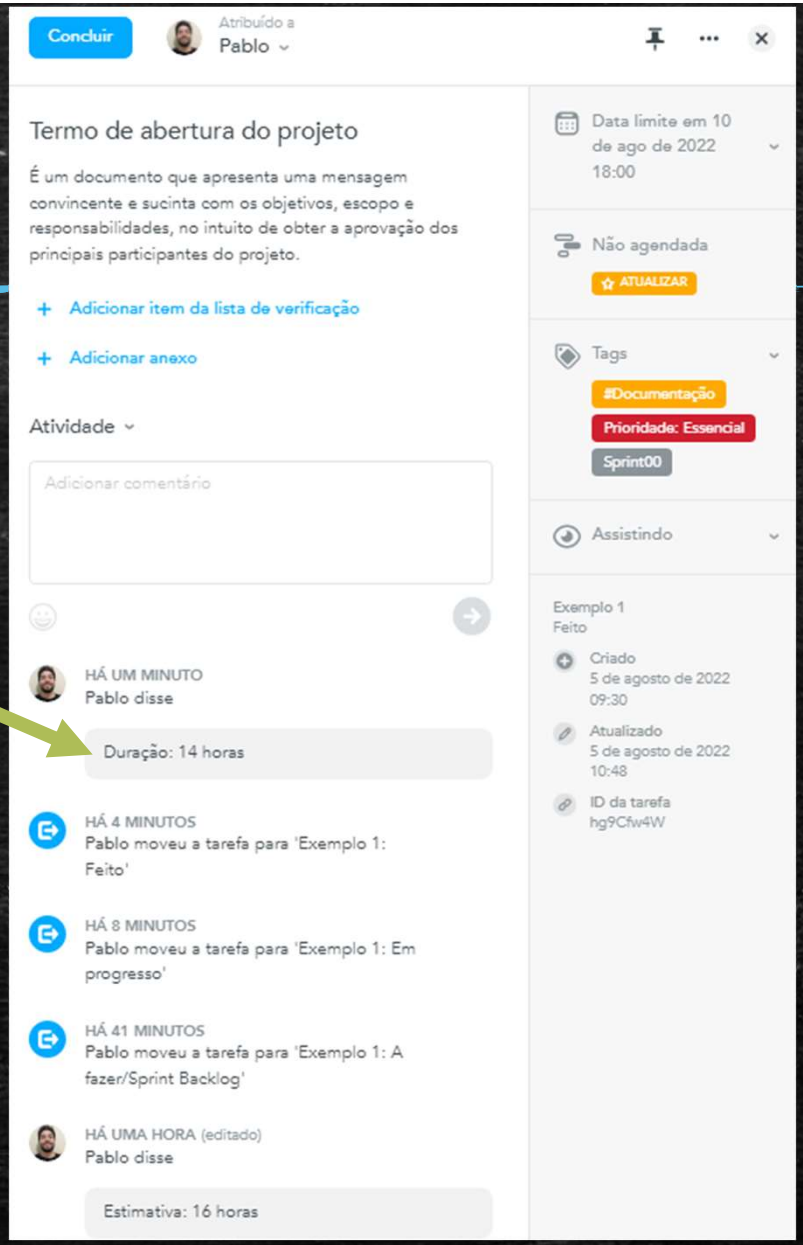
Simulando: mova Plano de gestão de riscos para "Em progresso"



Mais exemplos

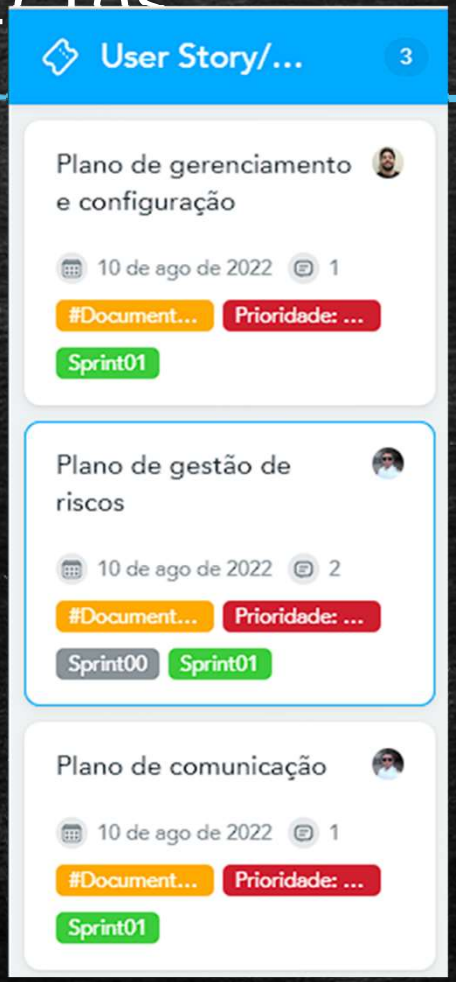
Ao enviar tarefas para seção "Feito" comente a duração dessa tarefa.

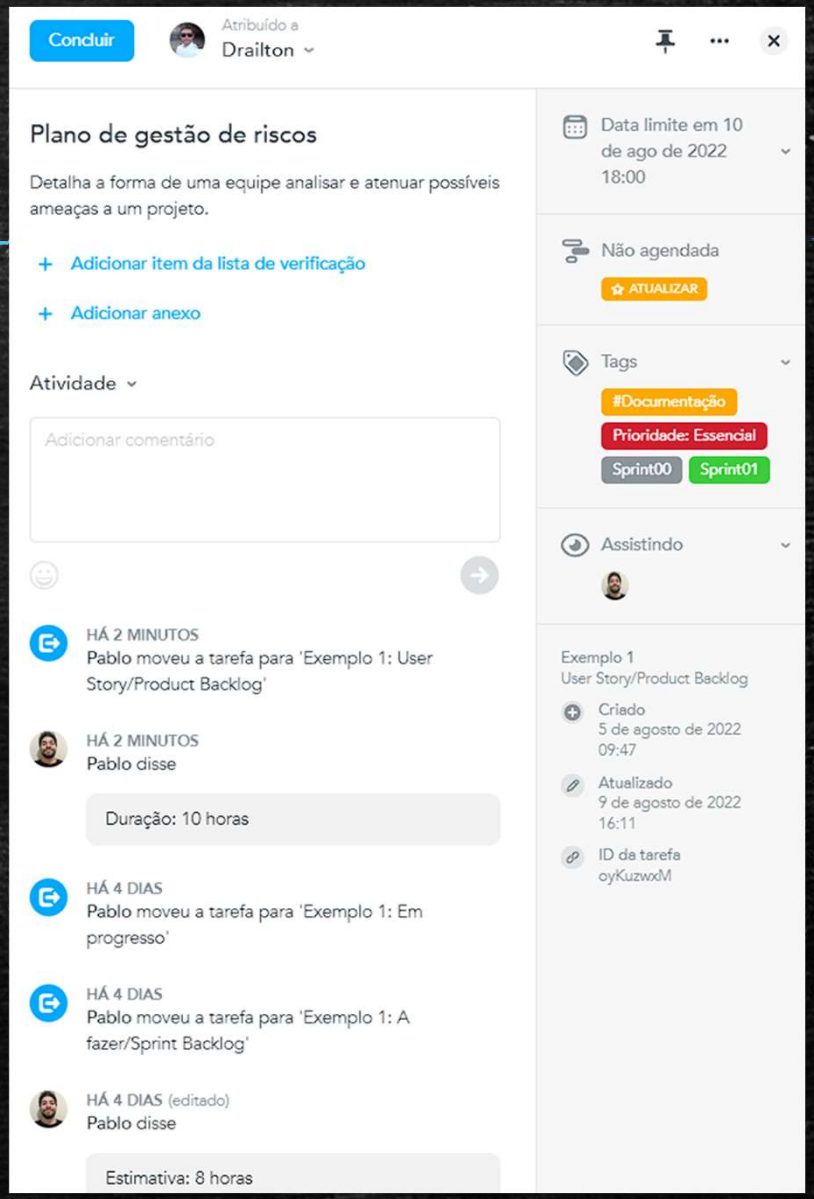
Isso ajuda a acompanhar o projeto alimentando o gráfico burndown



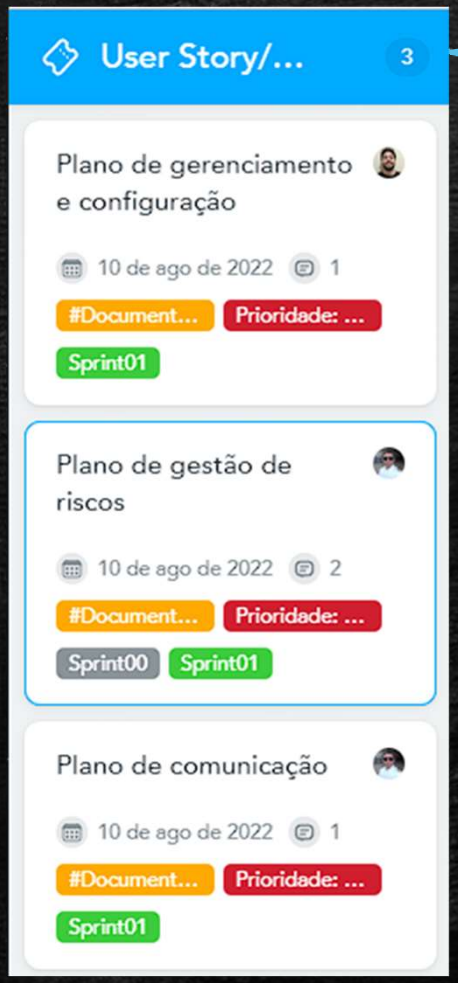
Mais exercícios

Pode ser que uma das tarefas não consiga ser concluída durante uma Sprint então ela ficará para a próxima Sprint.

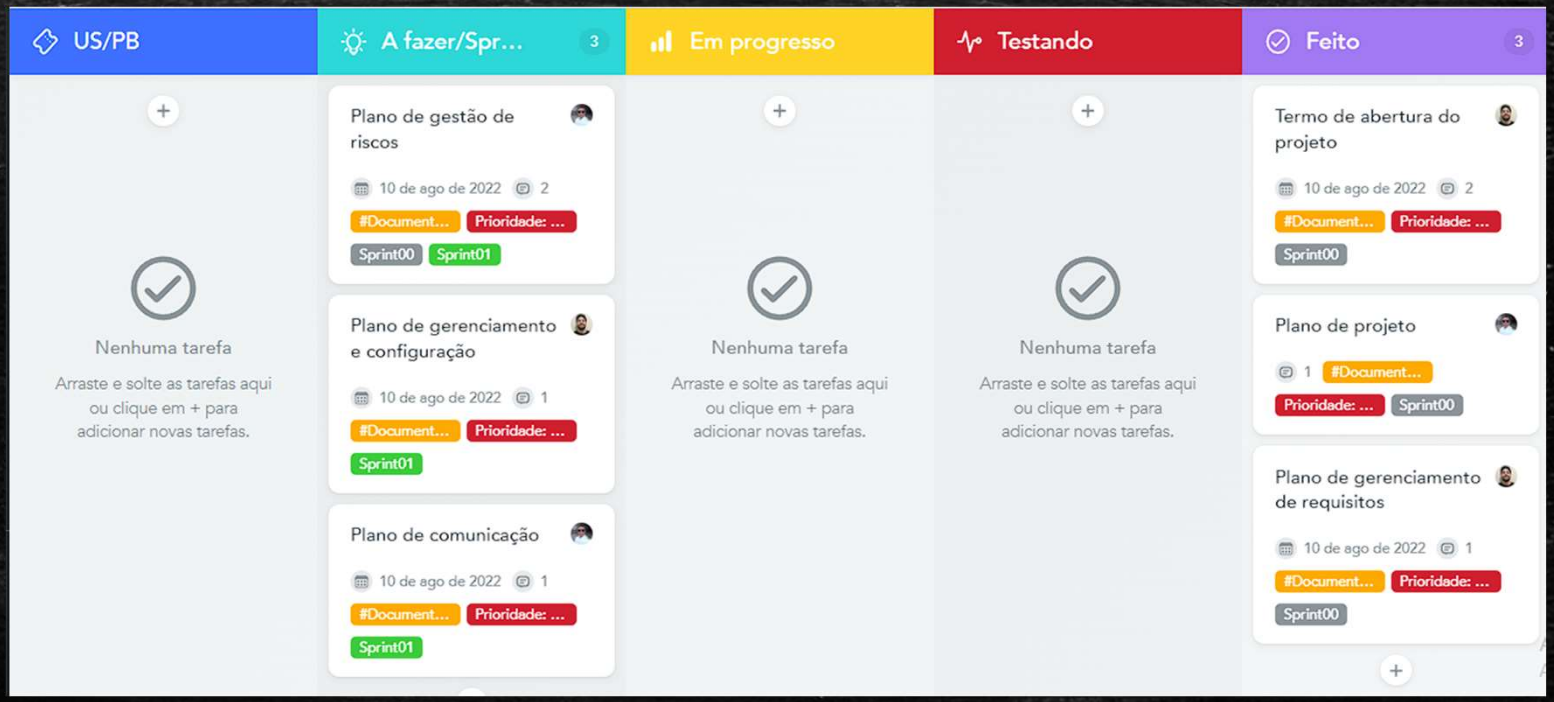




Mais exercícios

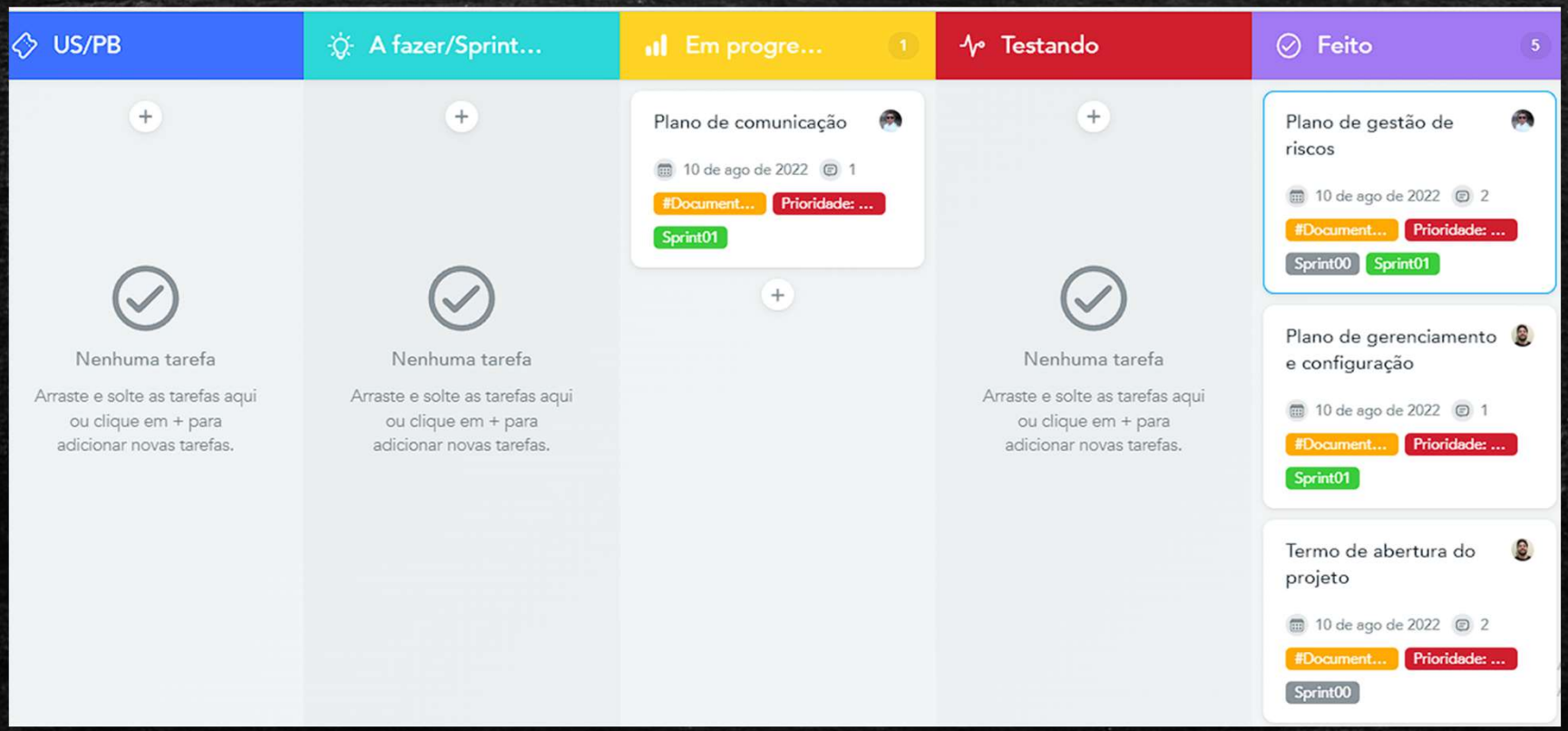


Simule a Sprint 01 inserindo comentários com a duração



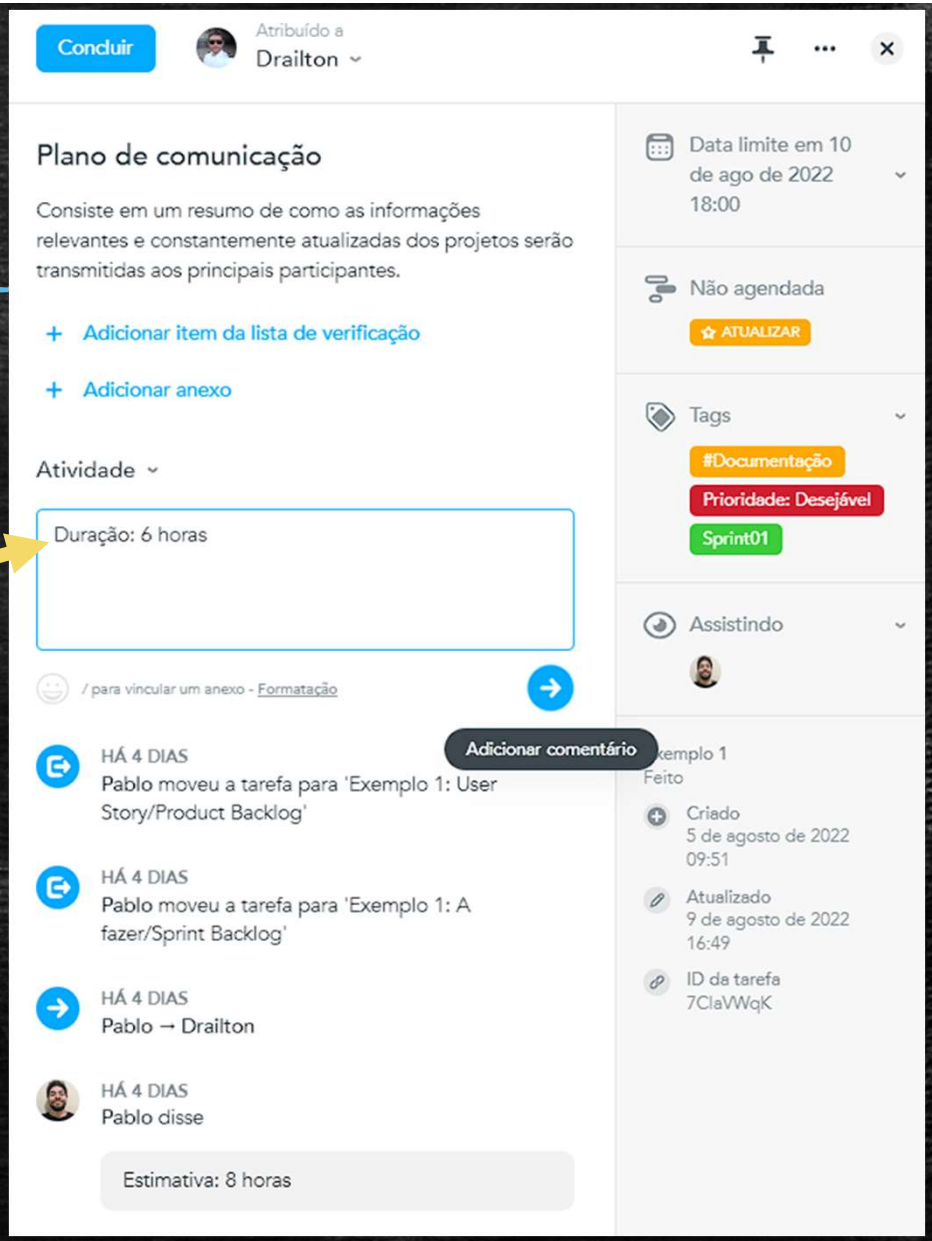
Mais exercícios

Simule a Sprint 01 inserindo comentários com a duração



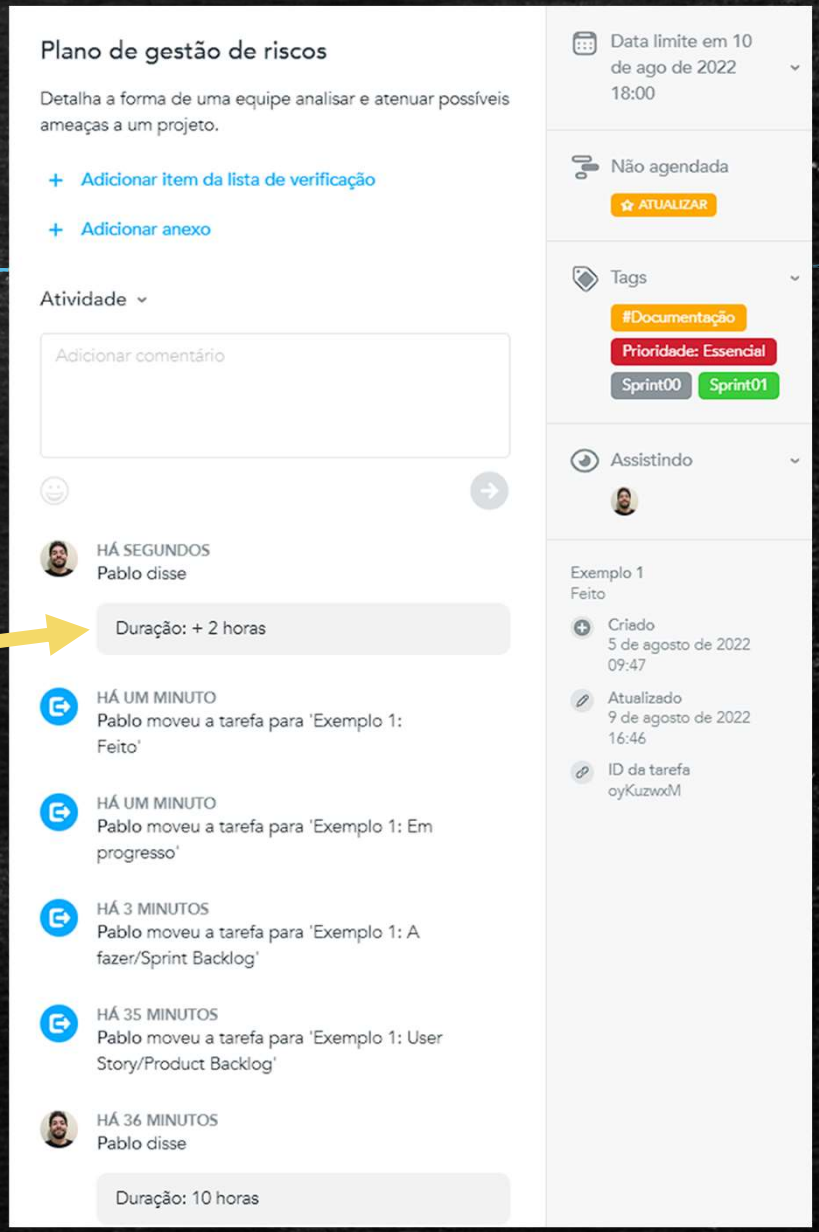
Mais exercícios

Adicione **duração** para cada tarefa quando tiver finalizado a Sprint ou a tarefa



Mais exercícios

No caso da tarefa que ficou em debito na Sprint passada adicione comentários com duração adicional que ocorreu nessa Sprint



Mais exercícios

Faça a US e as tarefas abaixo no Meister Task dividindo em 3 sprints

Crie um gráfico *burndown* (SM) simulando 4 dias como 4 semanas e 4 sprint.

Estime esforço total do projeto para os 4 "semanas" (64 horas documentação)

US: Como **gerente,** eu **gostaria de cadastrar os quartos do hotel** para disponibilizar mais ocupações.

- Incluir quarto na base de dados (4 horas)
- Excluir quarto na base de dados (4 horas)
- Consultar quarto cadastrados (2 horas)
- Editar quarto cadastrados (6 horas)

64+16=80 horas

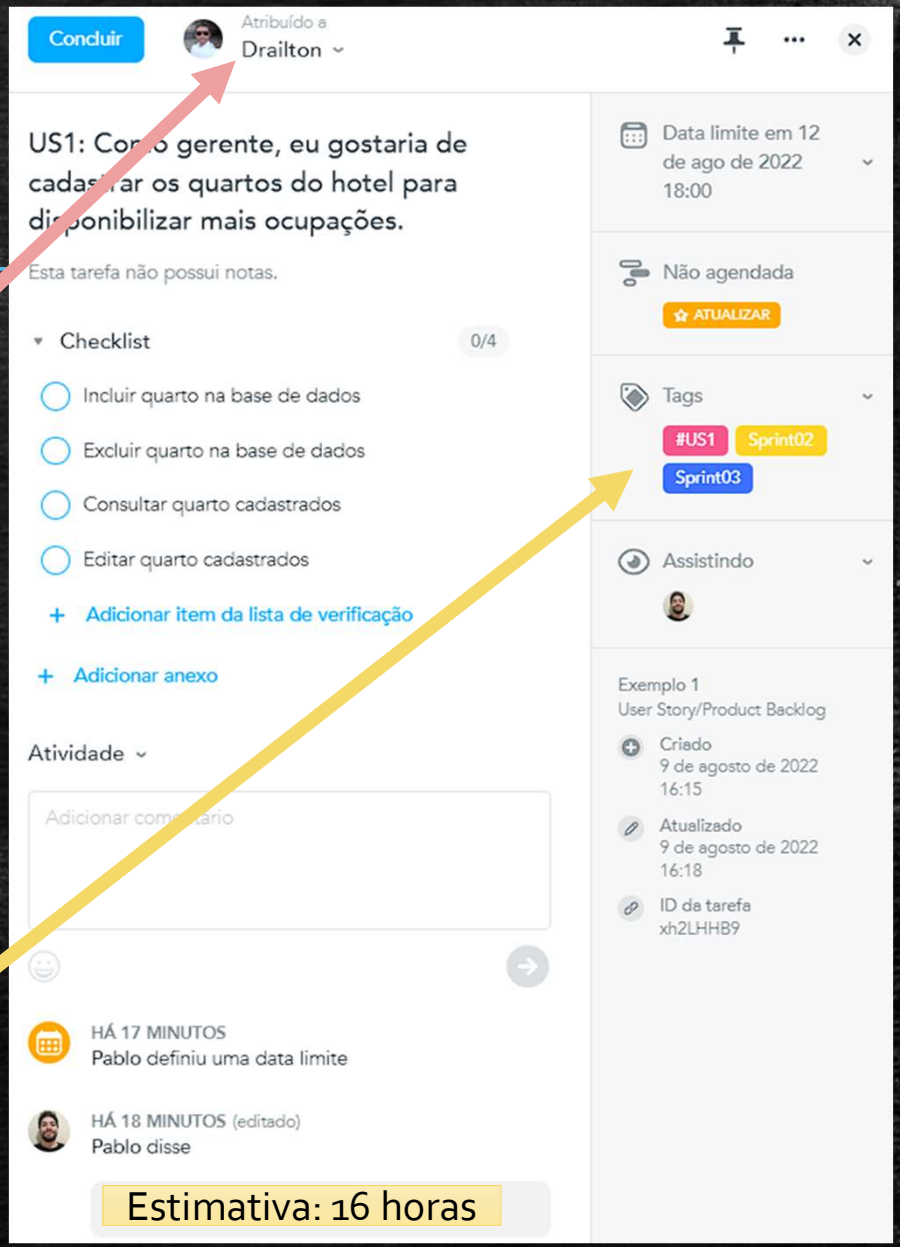
Mais exercícios

Não esqueça de atribuir de forma justa cada tarefa

Crie a US1 e seu checklist, assim como, as tarefas que foram geradas durante o refinamento.

US: Como **gerente**, eu **gostaria de cadastrar os quartos do hotel** para disponibilizar mais ocupações.

Não esqueça de adicionar tags para marcar a US e a Sprint

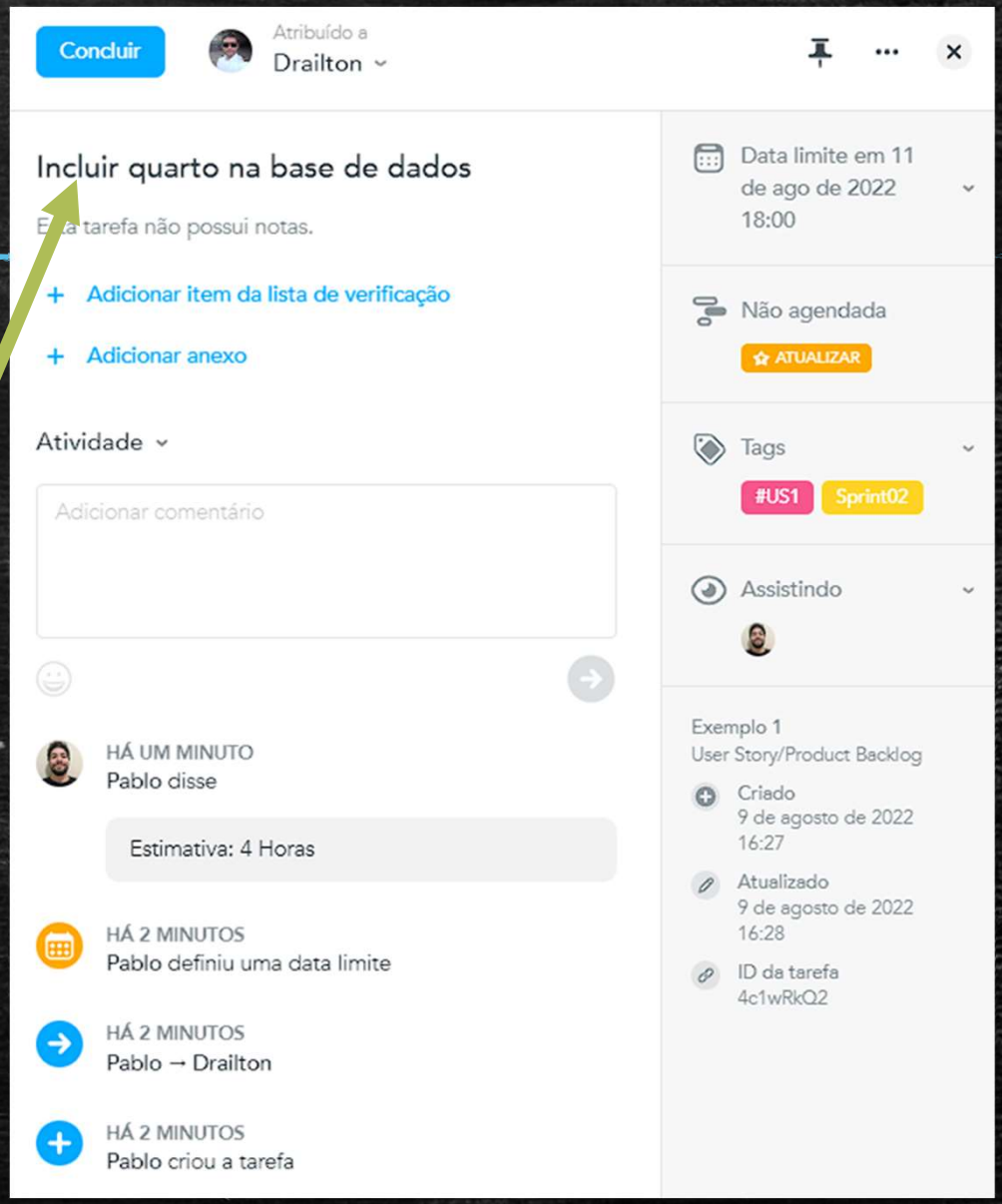


Estimativa: 16 horas

Mais exercícios

Adicione para cada item do checklist uma tarefa, usando tags para marcar as US e Sprint.

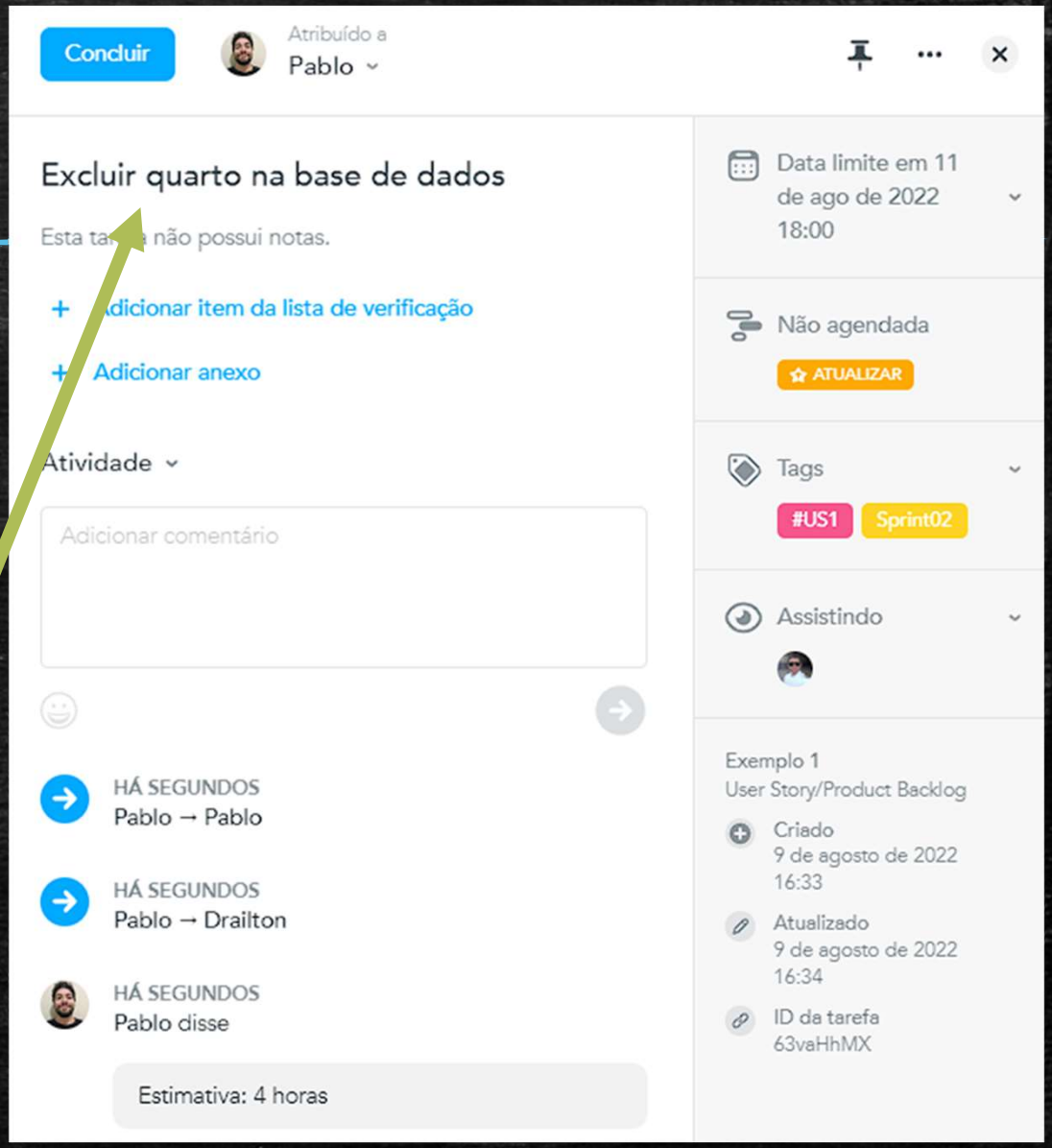
- Incluir quarto na base de dados
- Excluir quarto na base de dados
- Consultar quarto cadastrados
- Editar quarto cadastrados



Mais exercícios

Adicione para cada item do checklist uma tarefa, usando tags para marcar as US e Sprint.

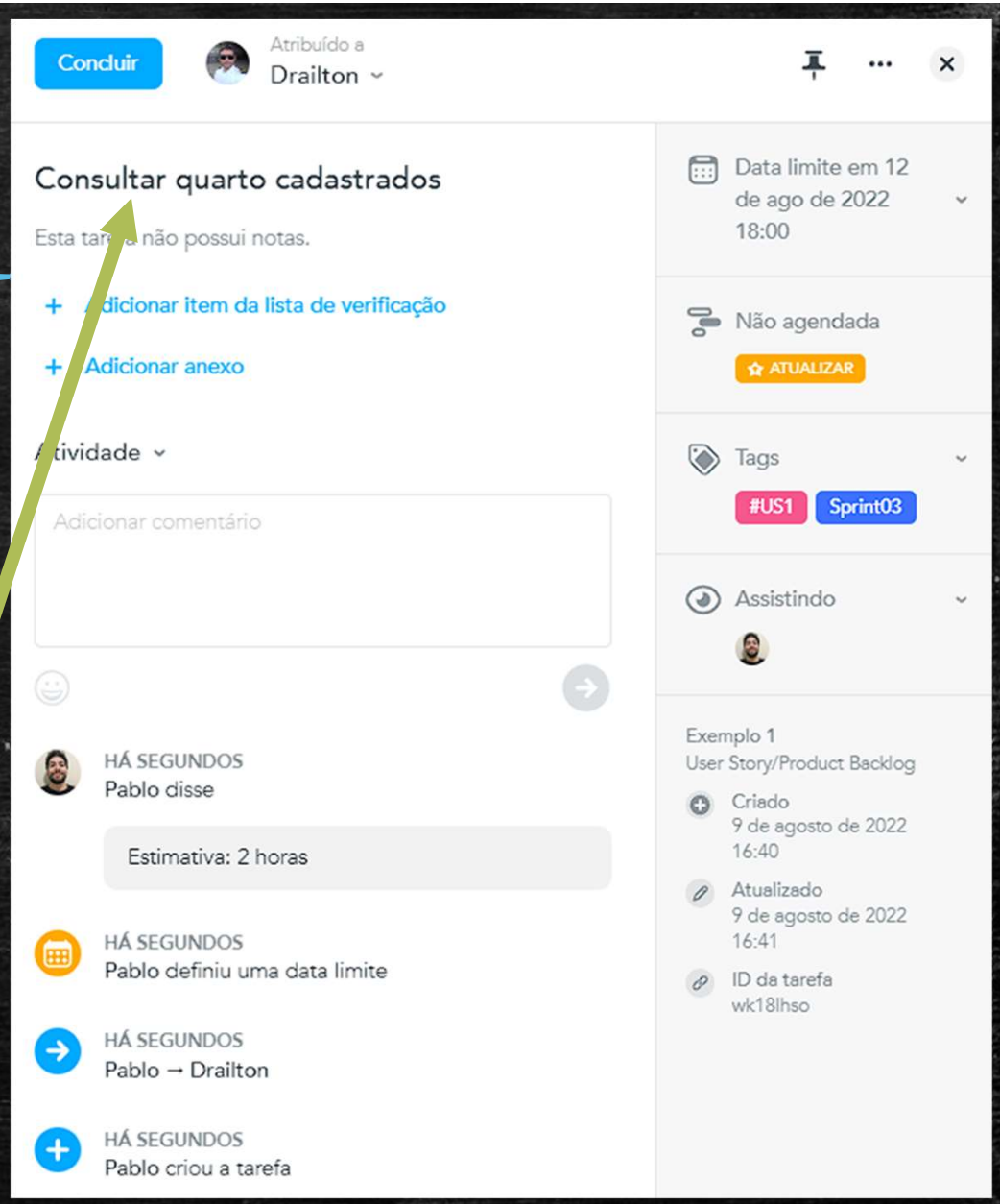
- Incluir quarto na base de dados
- Excluir quarto na base de dados
- Consultar quarto cadastrados
- Editar quarto cadastrados



Mais exercícios

Adicione para cada item do checklist uma tarefa, usando tags para marcar as US e Sprint.

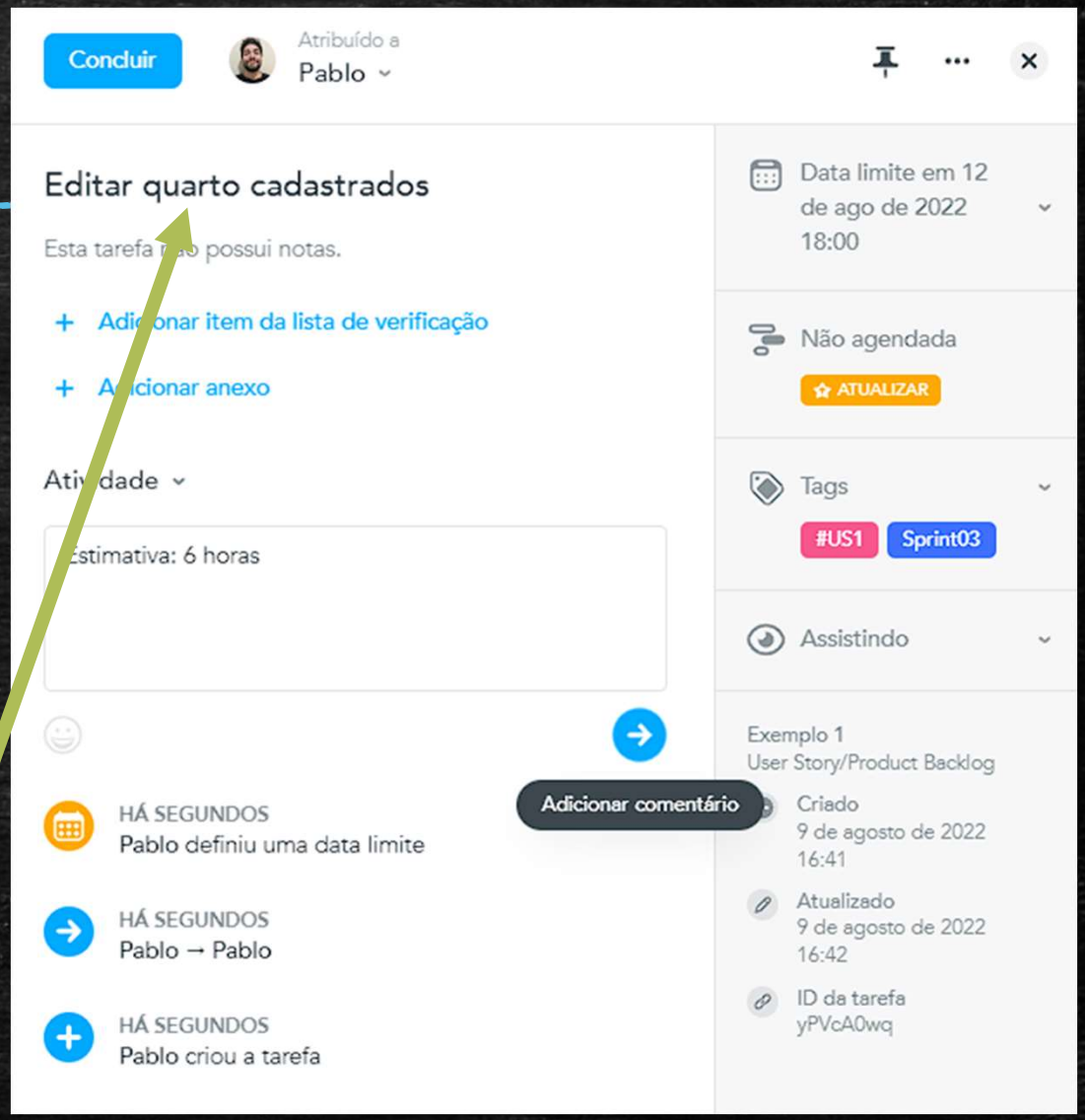
- Incluir quarto na base de dados
- Excluir quarto na base de dados
- Consultar quarto cadastrados
- Editar quarto cadastrados



Mais exercícios

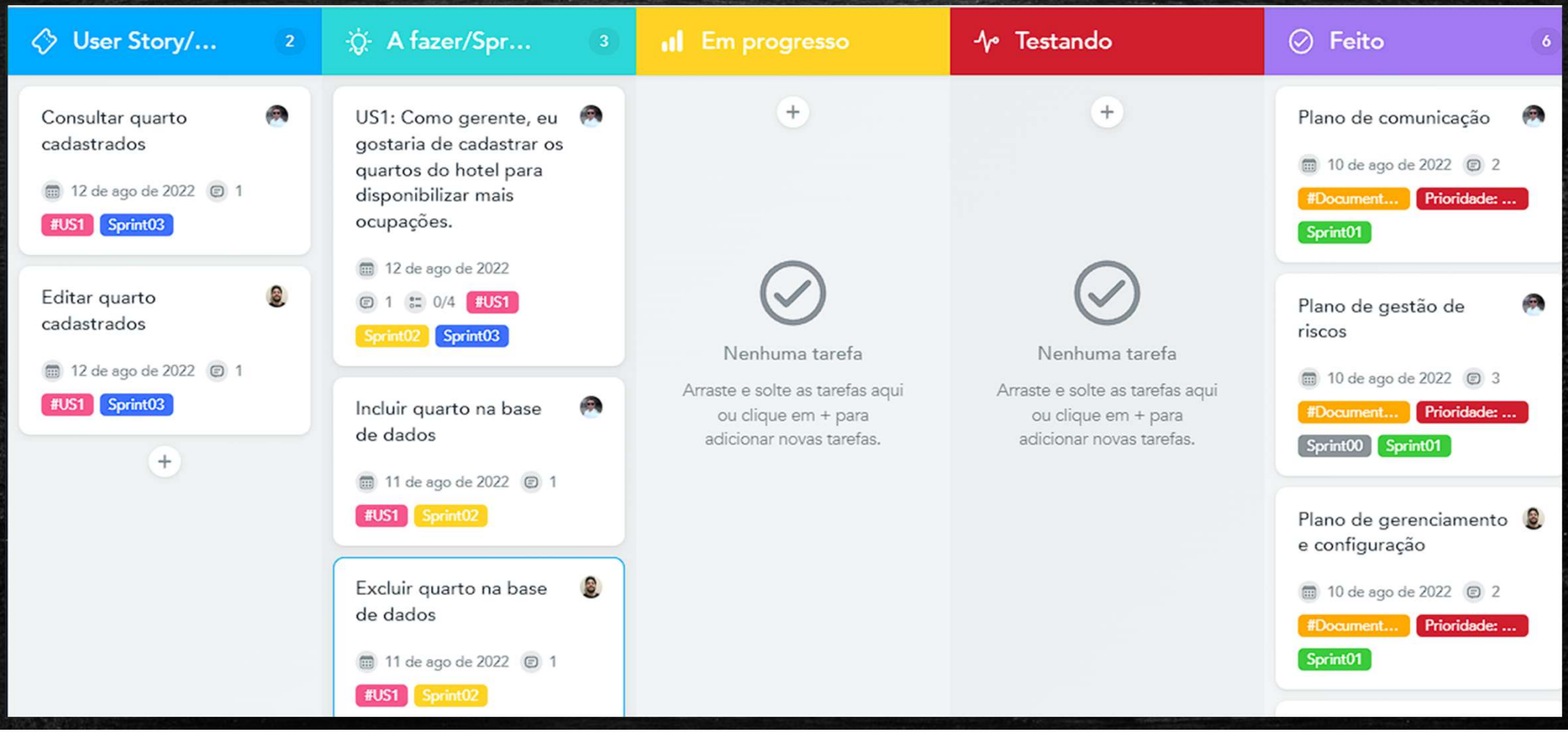
Adicione para cada item do checklist uma tarefa, usando tags para marcar as US e Sprint.

- Incluir quarto na base de dados
- Excluir quarto na base de dados
- Consultar quarto cadastrados
- Editar quarto cadastrados



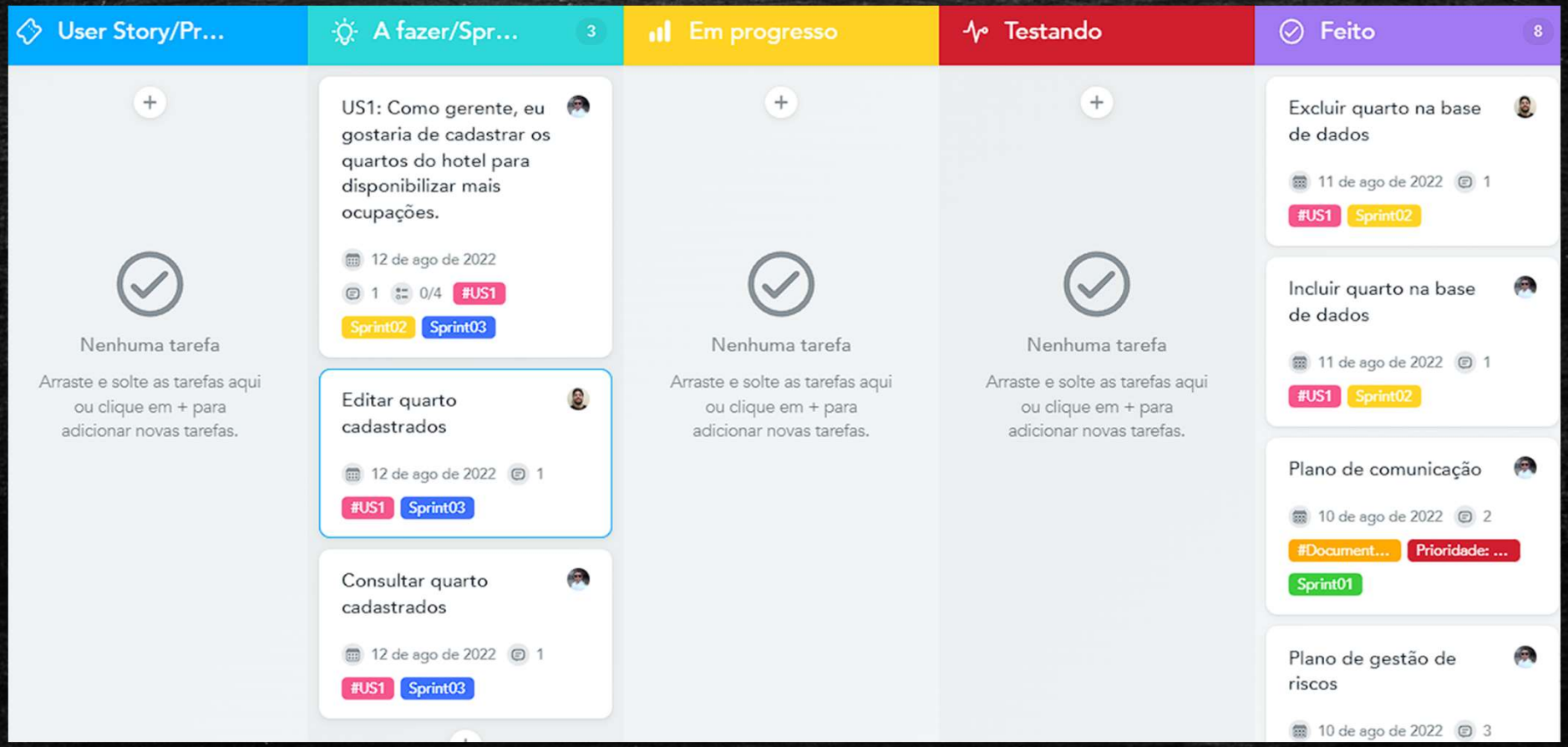
Mais exercícios

Simule a Sprint 02 e 03



Mais exercícios

Simule a Sprint 02 e 03



Mais exercícios

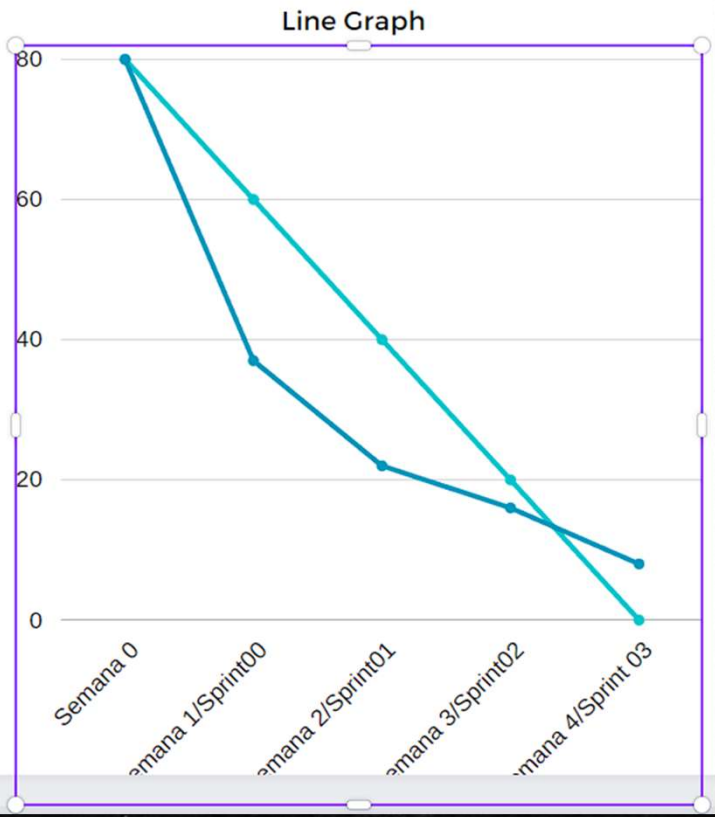
Gráfico Burndown

Dados | Configurações

	Horas (Est	Horas (Du
Semana 0	80	80
Semana 1/	60	37
Semana 2/	40	22
Semana 3/	20	16
Semana 4/	0	8

Limpar dados

Teve duração menor do que o esperado, sobrando 8 horas.



Exercícios para casa

- PO: Crie 3 ou 4 *User Story* sobre algum produto e componha o PB definindo prioridades para as historias.
- SM: Organize uma reunião estimando (Planning Poker) cada uma das *User Story* , bem como, definindo para quem será atribuída na Equipe Desenvolvedora.
- Devs: Refine as *User Story* em tarefas e estime um data para cada uma das tarefas.

PO: Define meta do PB, cria o projeto no Meister Task, adiciona a equipe e cria *User Story*.

SM: Deve montar o quadro Kanban no Meister Task e edita as *User Story.*

Devs: Deve criar as tarefas para cada *User Story (add tags)* no Meister Task.

Comparação com outros modelos

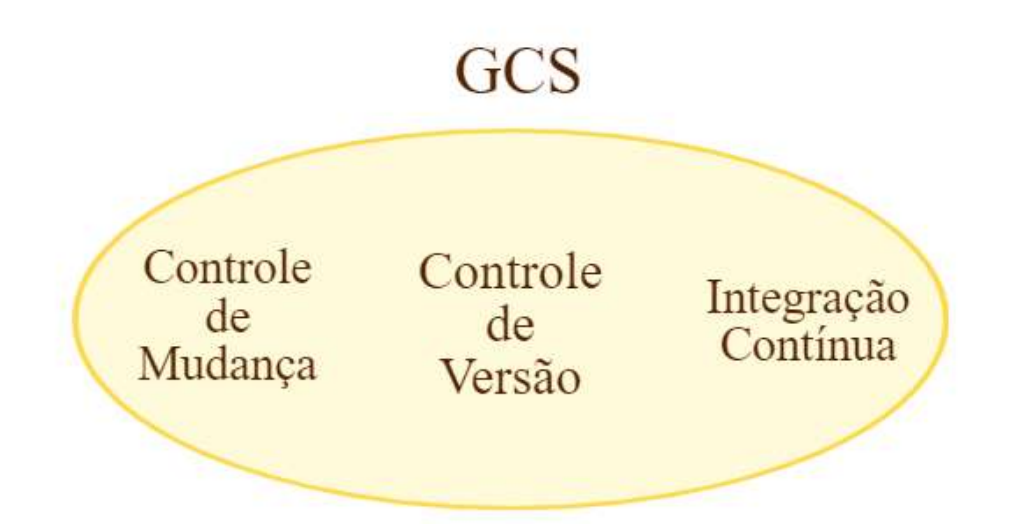
- **Com modelos incrementais**: Métodos ágeis diferem dos métodos incremental porque seus períodos de tempo são medidos em **semanas**, ao invés de meses, e a realização é efetuada de uma maneira altamente **colaborativa**.
- **Com modelo cascata:** No cascata a ênfase maior é no planejamento, entrega sem incrementos e rica documentação. O modelo cascata é o oposto dos modelos ágeis.
- **Com modelo balburdia**: Como os desenvolvedores que utilizam métodos ágeis frequentemente reavaliam os planos, enfatizam a comunicação e fazem o uso relativamente esparso de documentos, ocasionalmente levam as pessoas a confundirem isto com modelo balburdia. Equipes ágeis, contudo, seguem o processo definido e frequentemente de forma disciplinada e rigorosa.

Configuração

- “...é o **conjunto de itens** constituintes de um **sistema.**” (Hirama)
- Representa **o estado do conjunto de itens** que formam o sistema **em um determinado momento.**
- Auxilia em saber quais os **impactos ao projeto** quando há uma **solicitação de mudança.**
- Permite que se **gerar e monitorar novos *releases*** para incorporar uma nova funcionalidade.

Configuração

- Questões chaves:
  - **Por que o sistema mudou?** (Controle de Mudança)
  - **Quais foram as mudanças?** (Controle de Versão)
  - **O sistema continua funcionando depois das mudanças?** (Integração Contínua)



Gerenciamento de configuração e versão

- Tem o proposito de **manter o controle** sobre os **artefatos** de software (ex: Especificações de requisitos, de arquitetura e os códigos) e suas **versões**.
- Solução modularizada que **provê** a **estabilidade dos ambientes** de software e o **controle de arquivos.**
- "...é um conjunto de atividades desenvolvido para **administrar as mudanças** dentro do ciclo de vida" (Pressman)
- "...o desenvolvimento e a aplicação de procedimentos para **gerenciar um produto** de sistemas em desenvolvimento. (Sommerville)

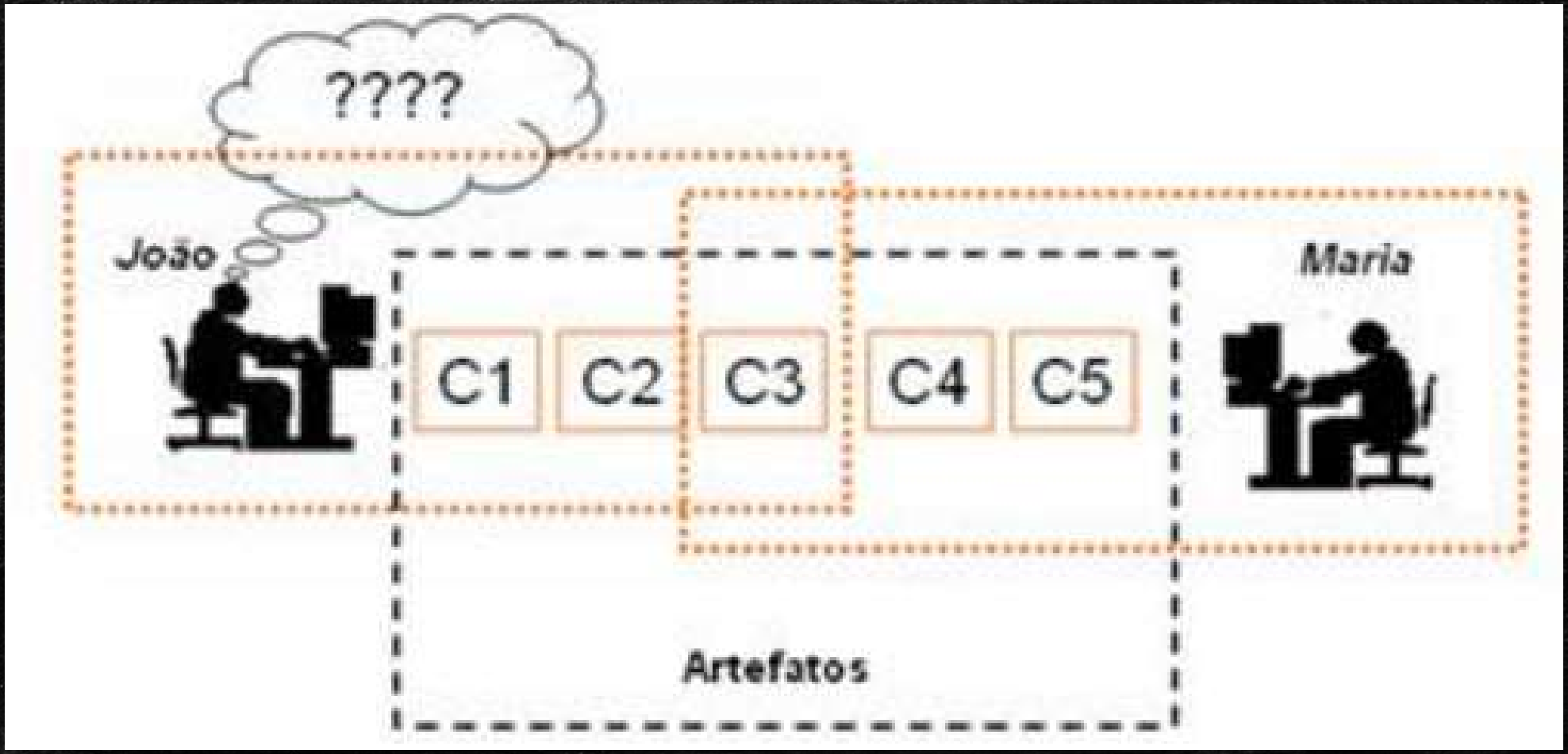
Gerenciamento de configuração e versão

- "...é o **processo de acompanhamento de diferentes versões** de componentes de software ou SCI e os sistemas em que esses componentes são usados." (Sommerville)
- Garante um ambiente seguro para que as diferentes mudanças realizadas pelos desenvolvedores **não interfiram uma nas outras.**
- Gerenciamento de *codelines* e *baselines*.

Gerenciamento de configuração e versão

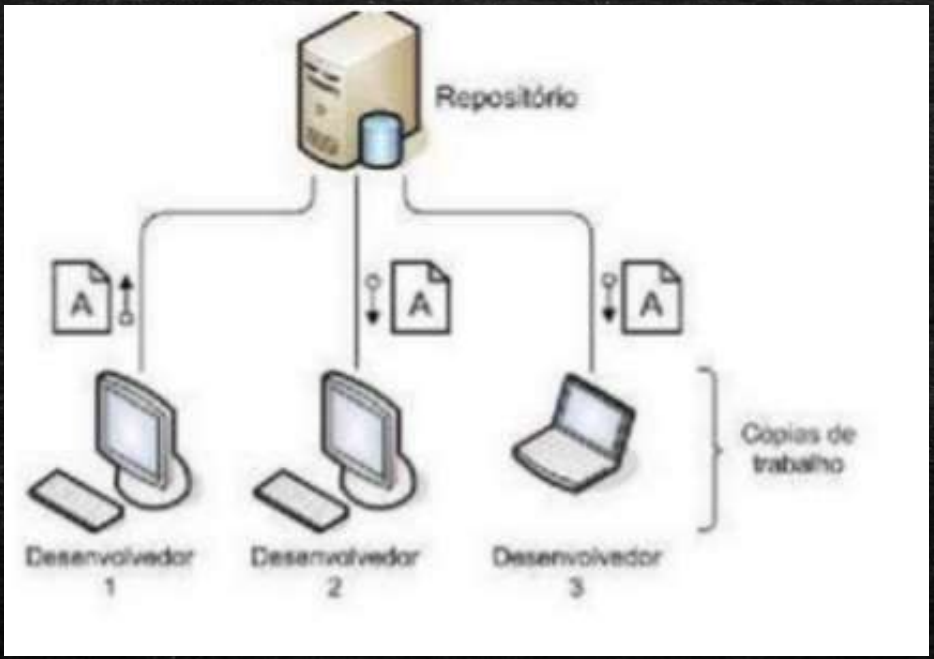
- Fortemente relaciona à **garantia de qualidade.**
  - Procura **garantir a integridade** dos produtos durante o desenvolvimento e a manutenção.
- Os produtos gerados no Gerenciamento de Configuração(GC) são chamados de **Itens de Configuração de Software** (SCI – *Software Configuration Item*).
  - Ex: Especificações de Requisitos ou de projetos ou de código e suas respectivas representações (Diagramas, Planos, Manuais, Programas, Relatórios...)

Gerenciamento de configuração e versão



Gerenciamento de configuração e versão

- **Repositório**: local de acesso controlado, onde as versões dos itens de configuração (SCI) são armazenados.



Gerenciamento de configuração e versão

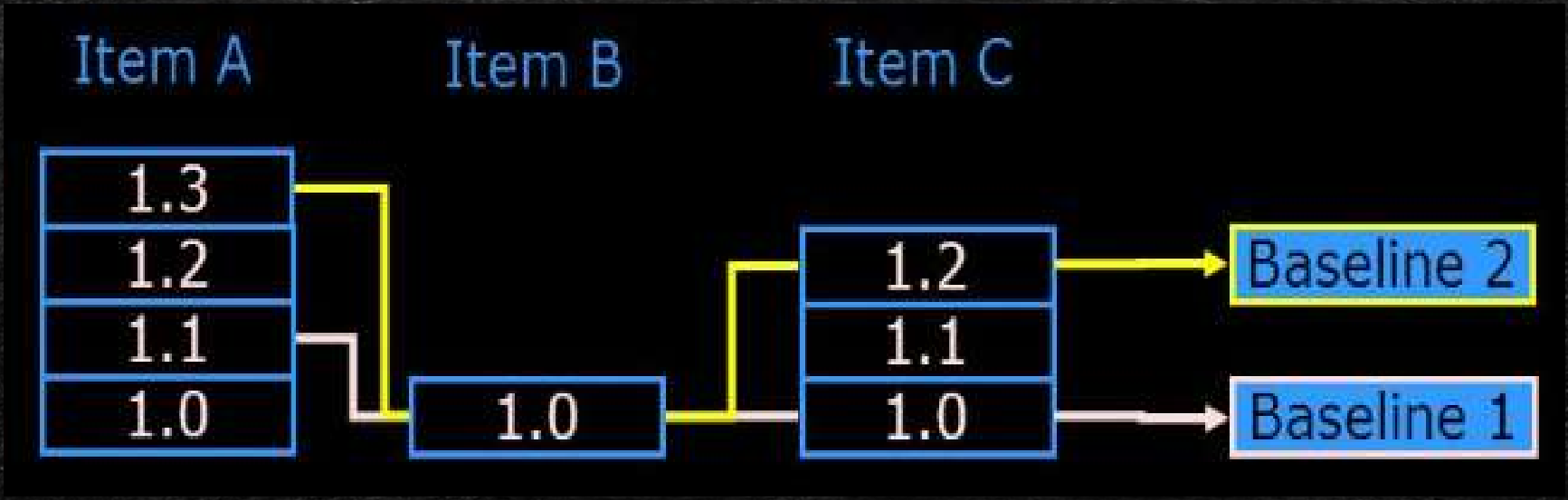
- **Controle de configuração**: o processo da garantia de que **versões** dos sistemas e componentes **sejam registradas e mantidas.**
- **Versão: uma instância de um item de configuração** que difere de outras instâncias desse item de alguma forma.
  - Possui um identificador único (Ex: componente 1.1, DER 4.1.2, Diagrama X.X.X)

Gerenciamento de configuração e versão

- ***Baseline***: é uma **coleção de versões** de componentes que compõem um sistema.
  - Uma **configuração do software** em determinado instante.
  - Só pode ser modificado através de procedimentos formais (Solicitação de Mudança).
  - Um artefato ou conjunto de artefatos só se torna um item de configuração depois que um *baseline* é estabelecido.

Gerenciamento de configuração e versão

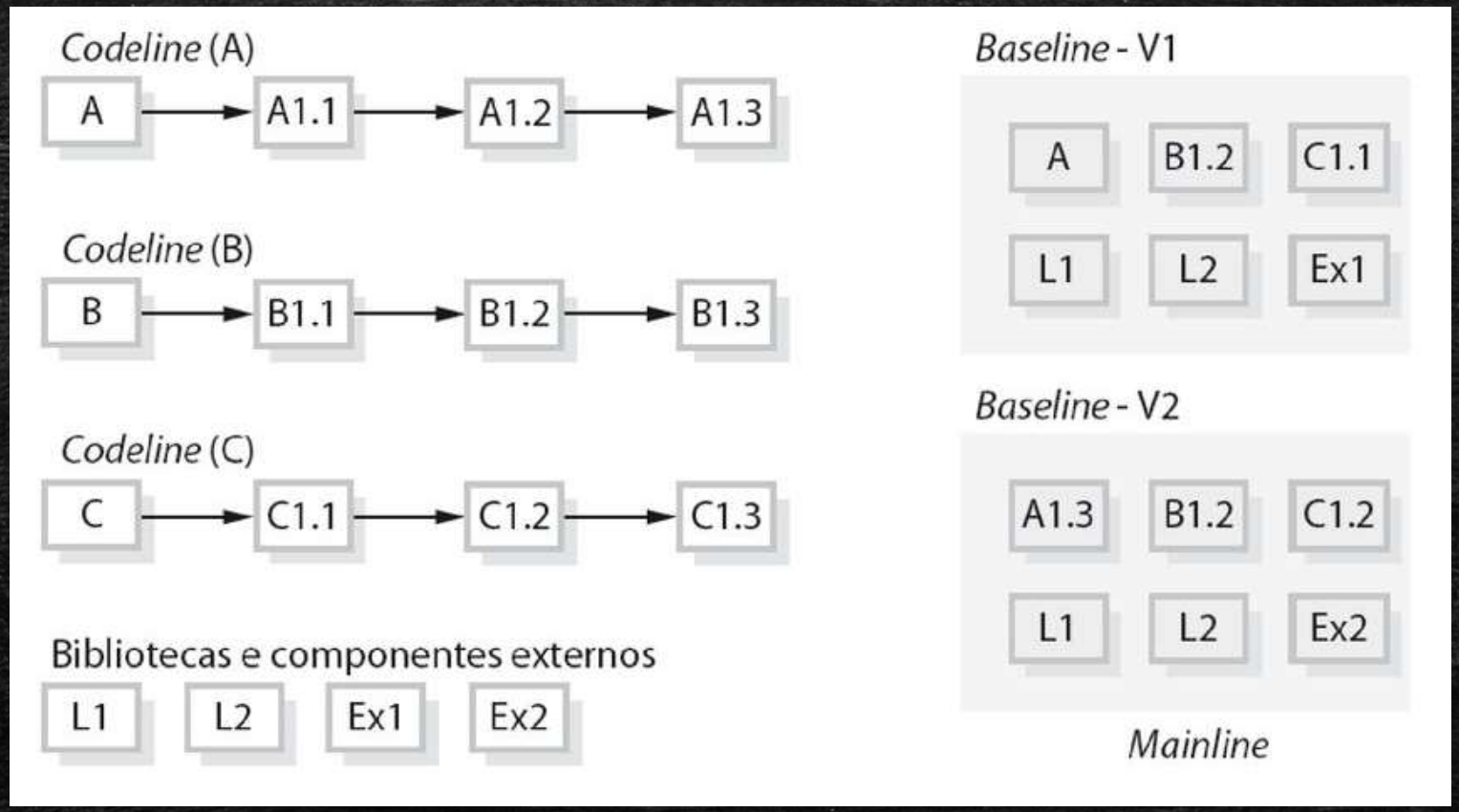
- *Baseline*.



Gerenciamento de configuração e versão

- ***Codeline:*** é um conjunto de versões de um componente de software e outros SCI dos quais esse componente depende.
- ***Mainline:*** trata-se de uma sequência de *baselines* que representam diferentes versões de um sistema.

Gerenciamento de configuração e versão



Gerenciamento de versões

- O que é "controle de versão" e porque eu deveria me importar?
  - É um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa lembrar versões específicas mais tarde.
  - Ex: Se você é um web designer e quer manter todas as versões de uma imagem, um sistema de controle de versão (VCS) é a coisa correta a ser usada.

Gerenciamento de versões

- **Sistema de Controle de Versões**: identificam, armazenam e controlam o acesso a diferentes versões de componentes. (Ex: Git, CVS e Subversion)
  - ...permite que os artefatos sejam obtidos, por meio de uma operação conhecida como *check-out*, modificados dentro do espaço de trabalho (*workspace*) do desenvolvedor e, depois, retornados ao repositório por meio de uma operação conhecida como *check-in*.

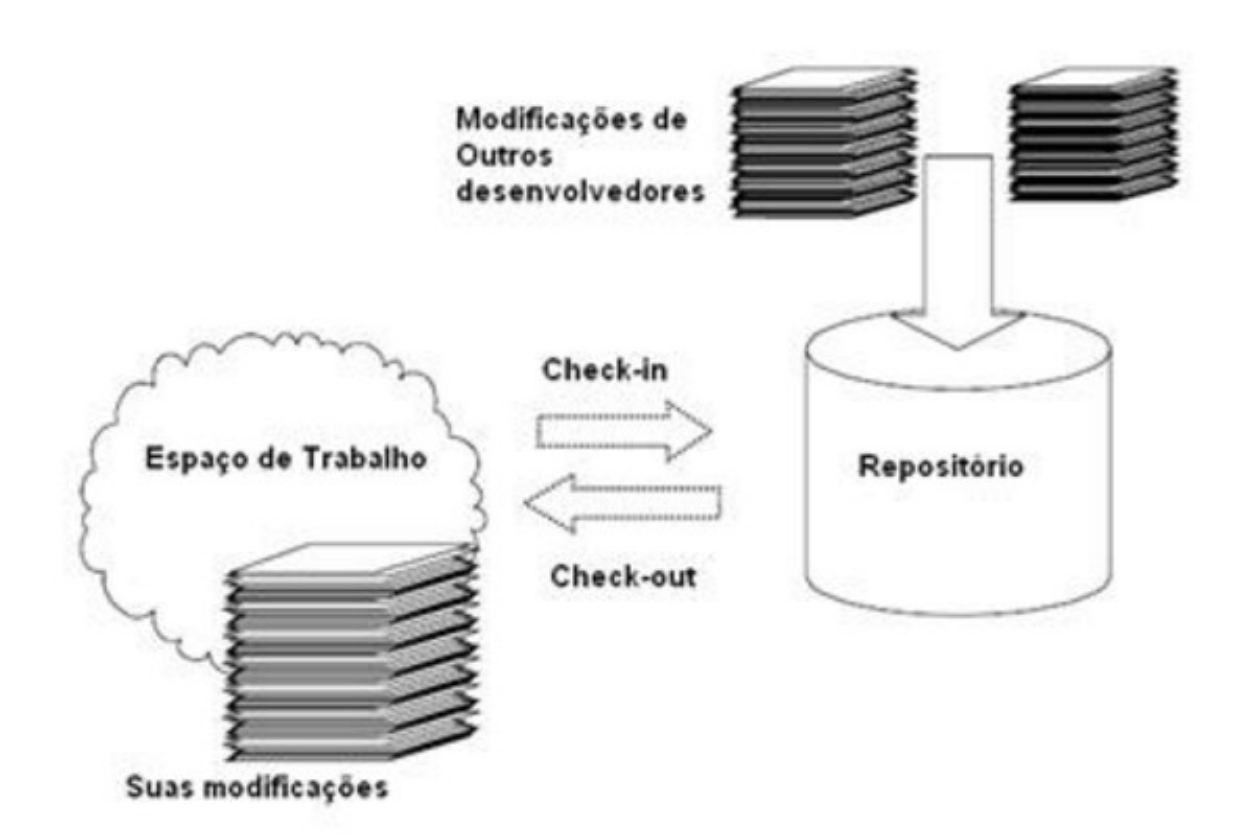
Gerenciamento de versões

- **Recursos oferecidos, geralmente, nos Sistema de Controle de Versões:**
  - Identificação de versão e release.
  - Gerenciamento de armazenamento.
  - Registro de histórico de mudanças.
  - Desenvolvimento independente.
  - Suporte aos projetos.

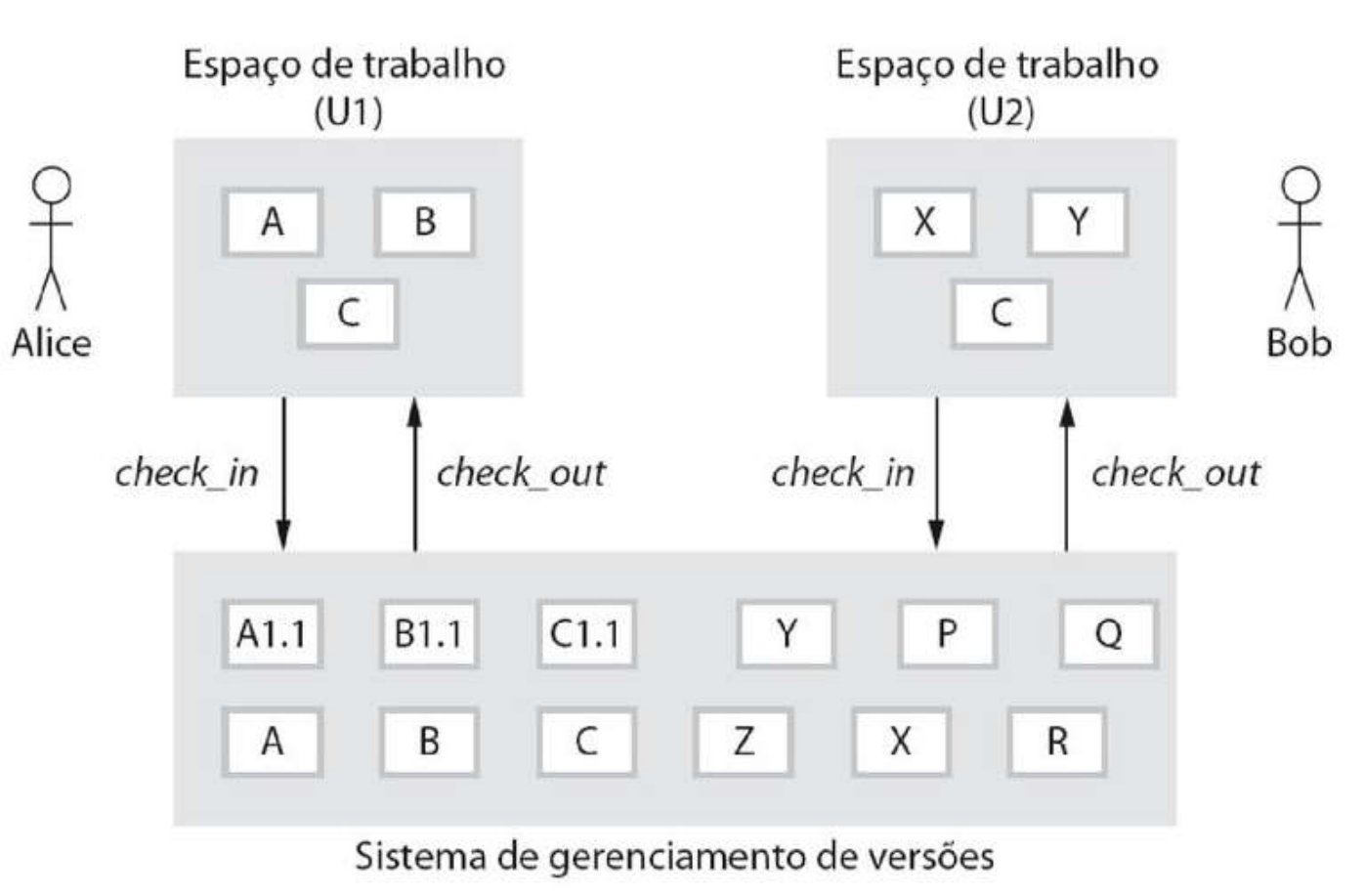
Gerenciamento de versões

- **Espaço de trabalho**: é uma área de trabalho em que o software pode ser modificado sem afetar outros desenvolvedores que possam estar usando ou modificando o software.
- ***Branching***: trata-se da criação de uma nova *codeline* de uma versão em uma *codeline* existente.
- ***Merging***: trata-se da criação de uma nova versão de um componente de software, incorporando versões separadas em diferentes *codelines*. Essas *codelines* podem ter sido criadas por um *branch* anterior de uma das *codelines* envolvidas

Gerenciamento de versões

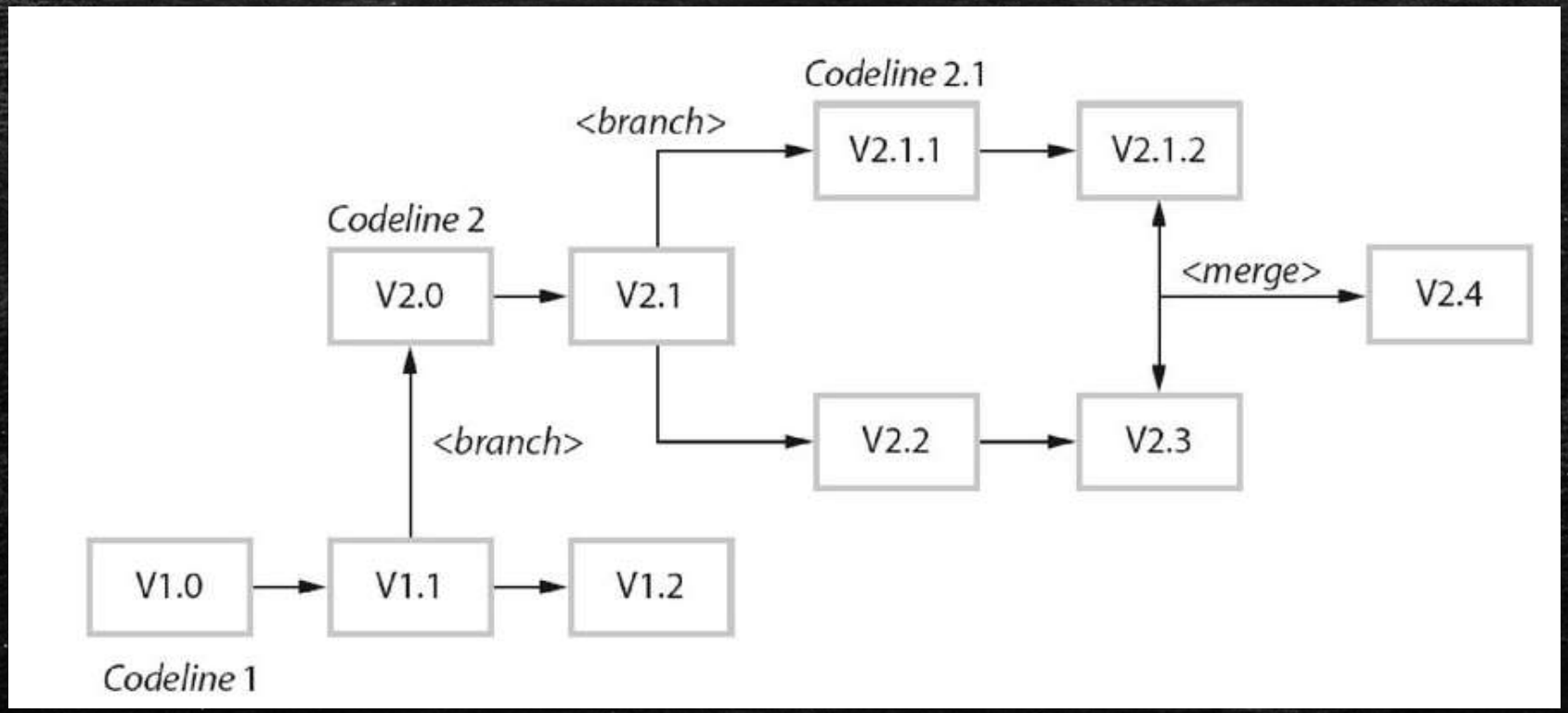


Gerenciamento de versões

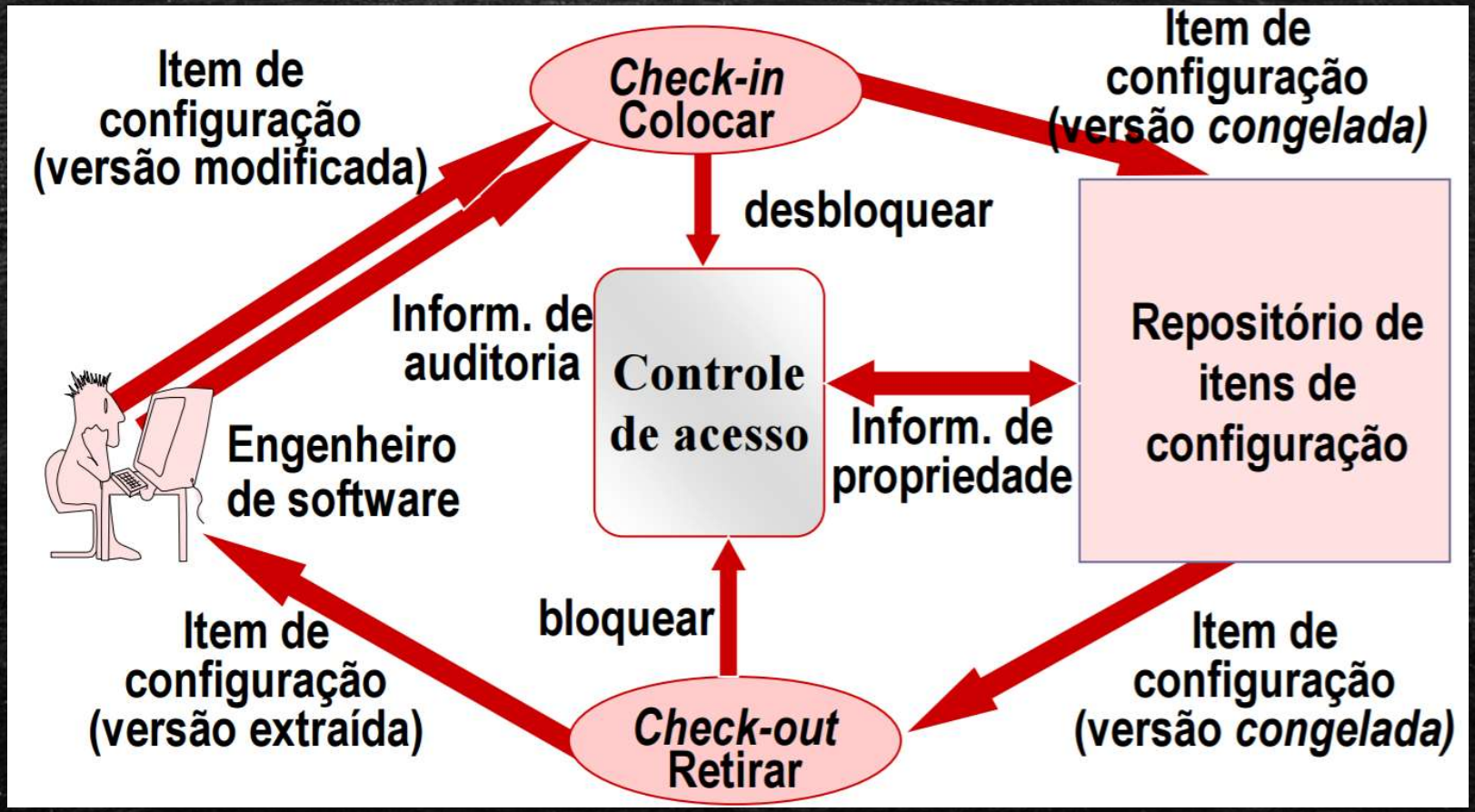


Gerenciamento de versões

- *Branching e merging*



Gerenciamento de versões



Sistemas de Gerenciamento de Configurações e Versões

- Alguns sistemas de versionamentos:
  - Concurrent Versions System – CVS
    - é um sistema de controle de versão que permite que se trabalhe com diversas versões de arquivos organizados em um diretório e localizados local ou remotamente, mantendo-se suas versões antigas e os logs de quem e quando manipulou os arquivos. (Wikipédia)
- Subversion – SVN Controla versão de diretórios, cópias e mudanças de nome.
  - é um sistema de controle de versão desenhado especificamente para ser um substituto moderno do CVS. (Wikipédia)

Sistemas de Gerenciamento de Configurações e Versões

- Alguns sistemas de versionamentos:
  - Mercurial
    - Alta performance e escalabilidade, descentralização, desenvolvimento colaborativo distribuído, controle de arquivos textuais e binários de forma robusta, e operações avançadas de ramos (branches) e mesclagem (merges).

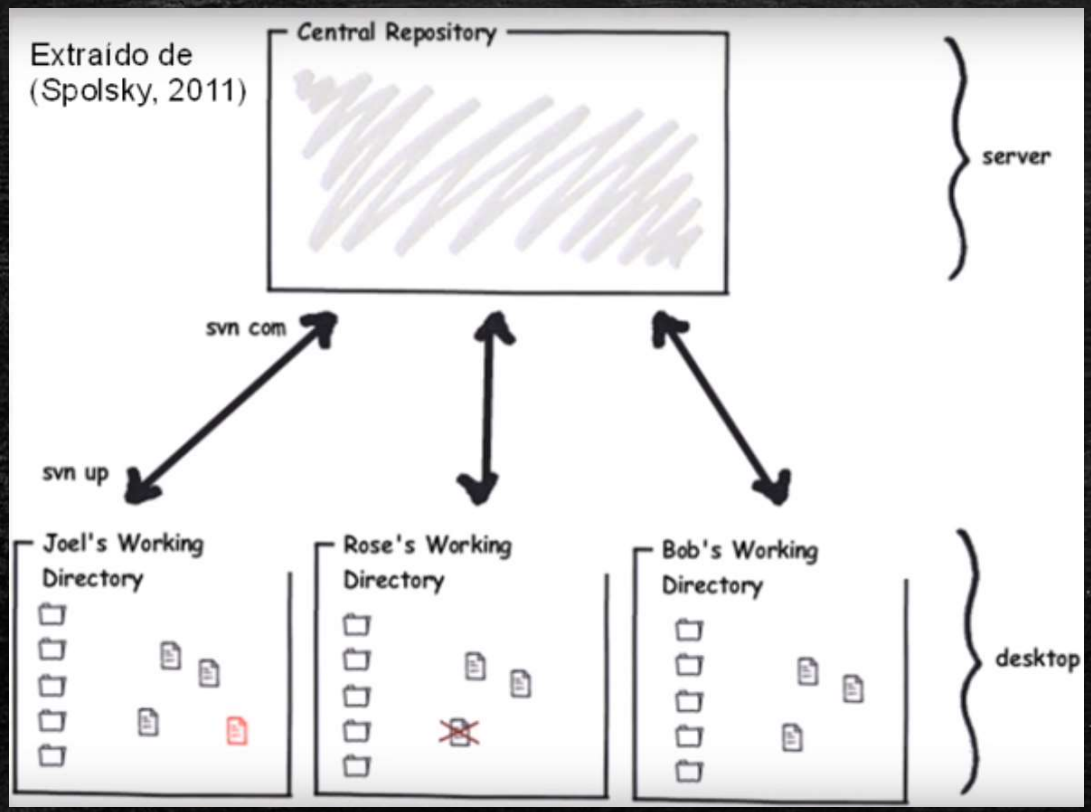
      Branch: ponteiro móvel que aponta para algum commit.

Sistemas de Gerenciamento de Configurações e Versões

- Alguns sistemas de versionamentos:
  - Git
    - Um sistema de controle de versão, ou VCS, que rastreia o histórico de alterações à medida que pessoas e equipes colaboram em projetos juntos.
      - À medida que os desenvolvedores fazem alterações no projeto, qualquer versão anterior do projeto pode ser recuperada a qualquer momento.

Sistemas de Gerenciamento de Configurações e Versões

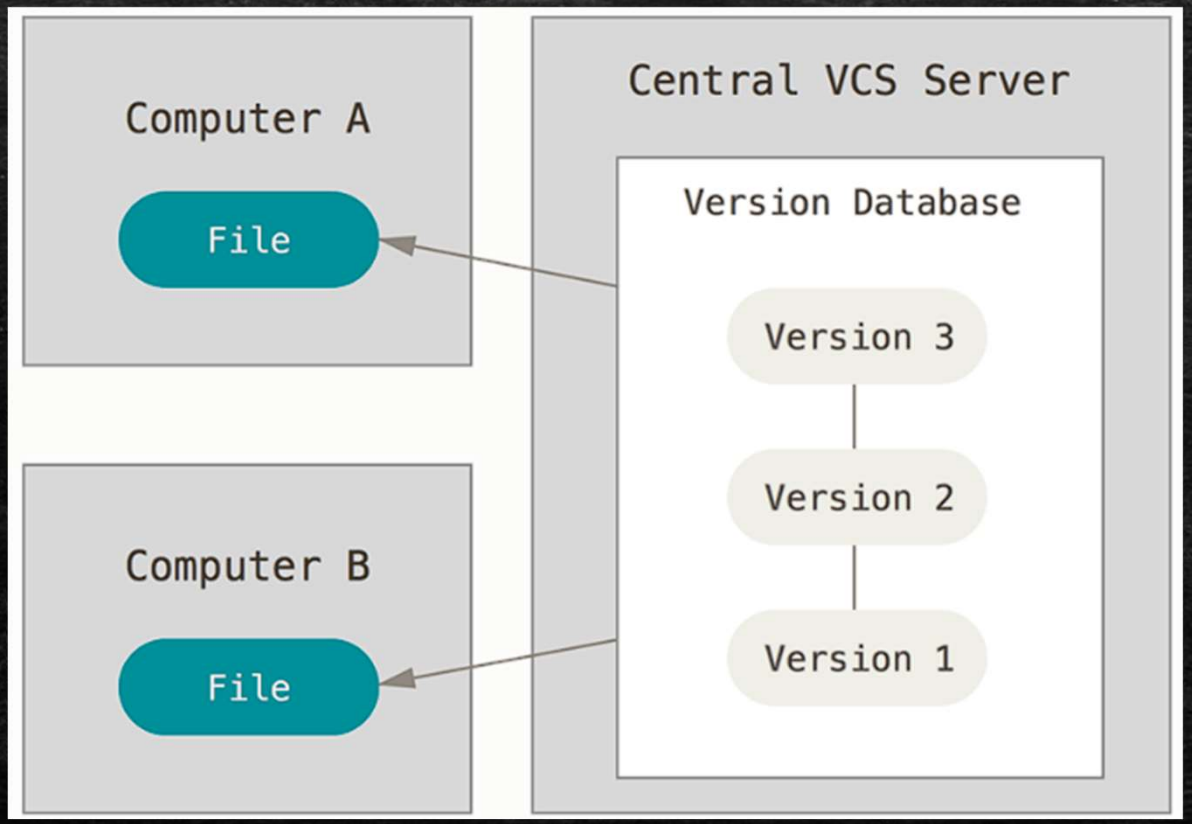
- Arquitetura Centralizada



Têm um único servidor que contém todos os arquivos de controle de versão e um número de clientes que usam arquivos a partir desse lugar central.

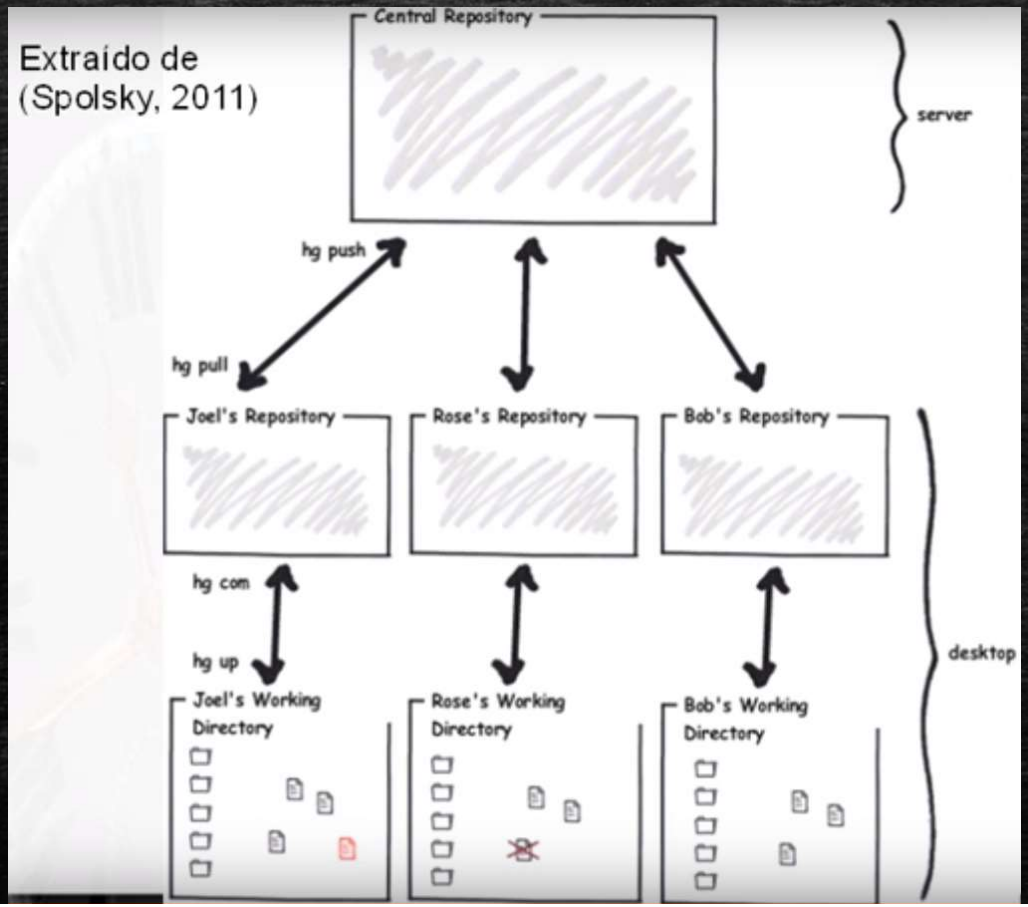
Sistemas de Gerenciamento de Configurações e Versões

- Arquitetura Centralizada



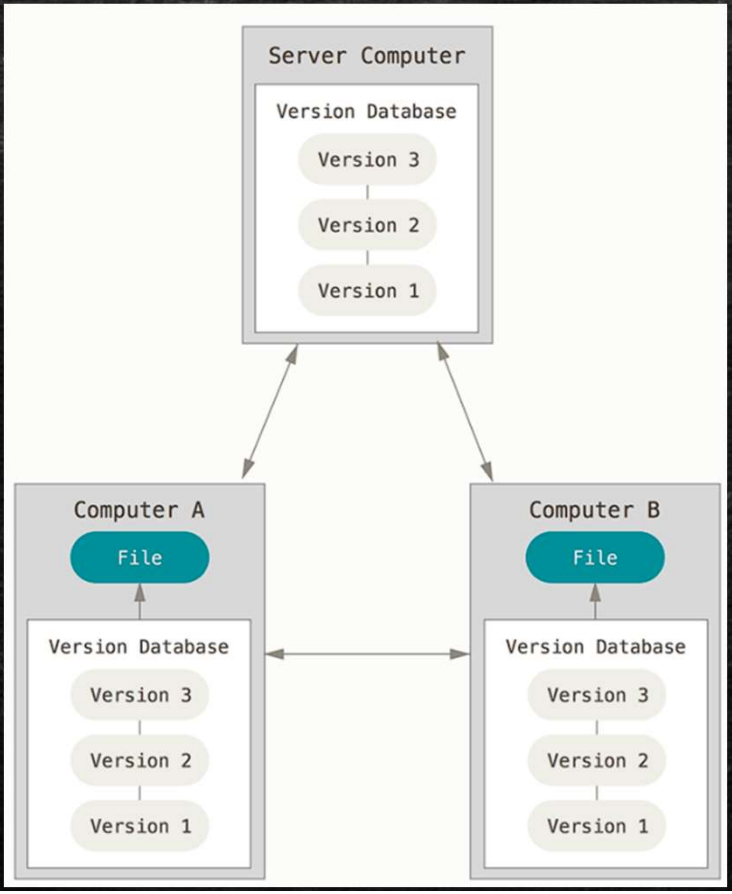
Sistemas de Gerenciamento de Configurações e Versões

- Arquitetura Descentralizada



Sistemas de Gerenciamento de Configurações e Versões

- Arquitetura Descentralizada



Git

- Em 2005, a relação entre a comunidade Linux e a BitKeeper quebrou em pedaços, e a ferramenta passou a ser paga.
  - Isto alertou a comunidade Linux (e especialmente Linux Torvalds) a desenvolver a sua própria ferramenta baseada em lições aprendidas ao usar o BitKeeper.
- Trata seus dados mais como um conjunto de imagens de um sistema de arquivos em miniatura.
- trata seus dados mais como um fluxo de estado dos arquivos.

Git

- Metas do Git:
  - Velocidade
  - Projeto simples
  - Forte suporte para desenvolvimento não-linear (milhares de ramos paralelos)
  - Completamente distribuído
  - Capaz de lidar com projetos grandes como o núcleo o Linux com eficiência (velocidade e tamanho dos dados)

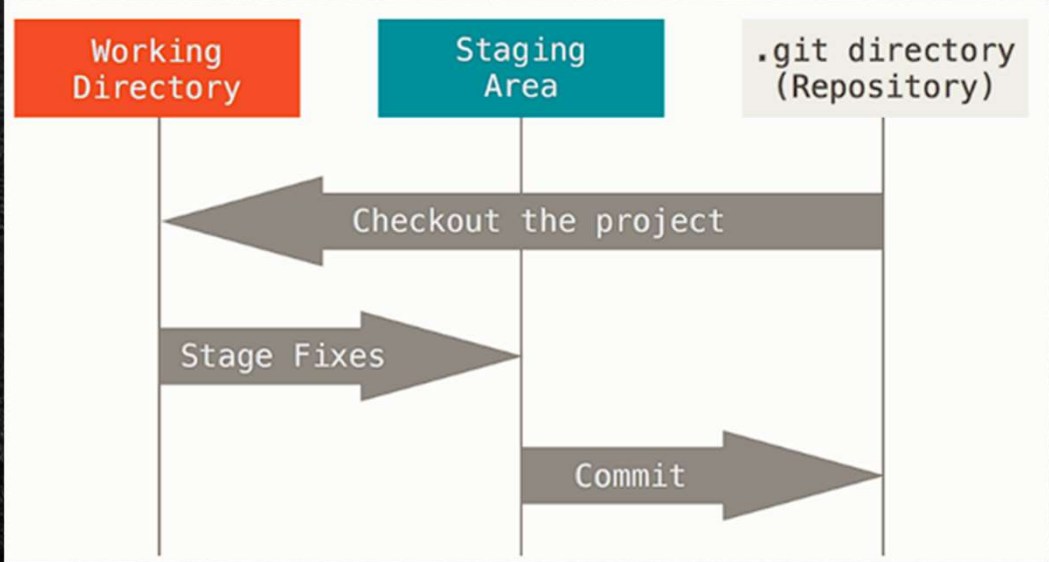
Git

- Podemos revisar o histórico do projeto para descobrir:
  - Quais mudanças foram feitas?
  - Quem fez as mudanças?
  - Quando foram feitas as mudanças?
  - Por que as mudanças foram necessárias?

Git

- Quase todas as operações são locais.
- Tem Integridade.
  - É impossível mudar o conteúdo de qualquer arquivo ou pasta sem que Git saiba.
- Geralmente Somente Adiciona Dados.
- Os Três Estados: committed, modificado (modified) e preparado (staged).

Git



Committed significa que os dados estão armazenados de forma segura em seu banco de dados local.

Modificado (modified) significa que você alterou o arquivo, mas ainda não fez o **commit** no seu banco de dados.

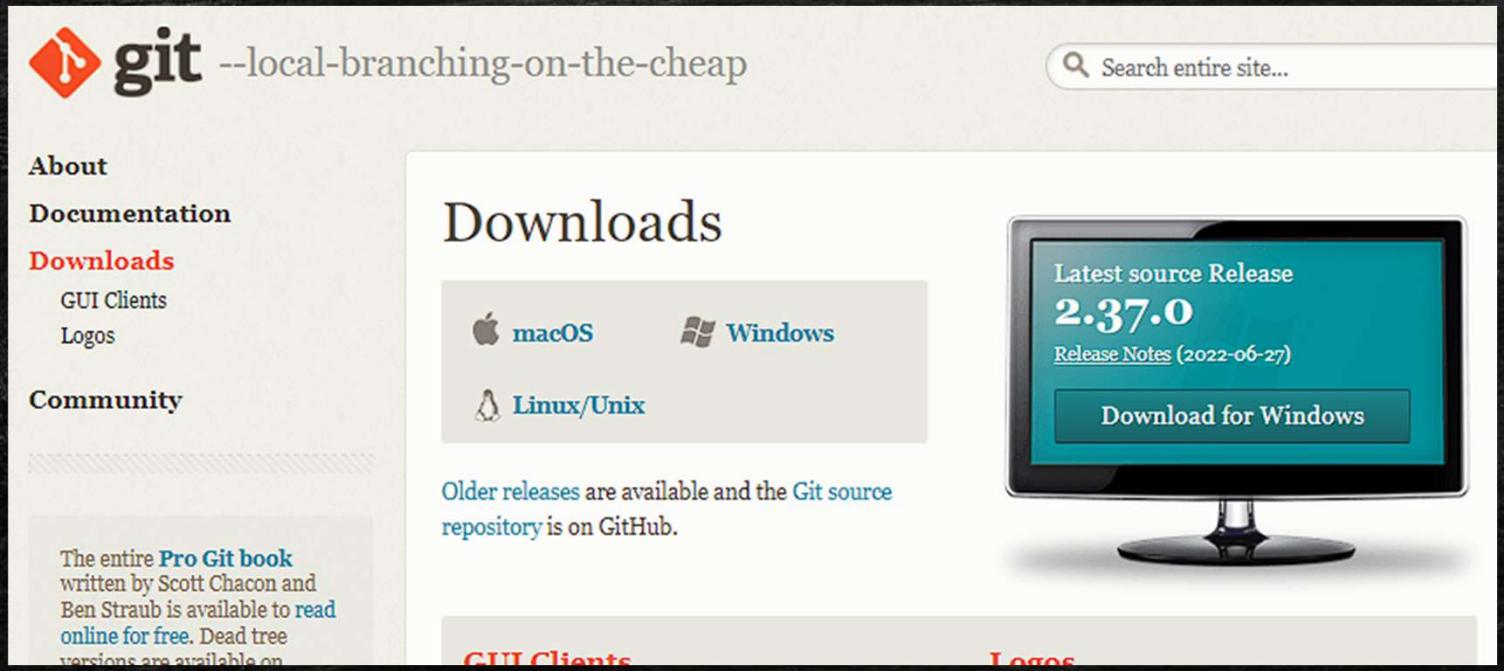
Preparado (staged) significa que você marcou a versão atual de um arquivo modificado para fazer parte de seu próximo commit.

Github

- Serviço de Web compartilhado para projetos que utilizam o Git para versionamento.
- Tratado análogo à uma rede social no qual se pode disponibilizar códigos.

Instalando o Git

- Acesse:
  - https://git-scm.com/downloads



Escolha a versão de acordo com seu SO

Configurando o Git

- O Git vem com uma ferramenta chamada git config que permite ver e atribuir variáveis de configuração que controlam todos os aspectos de como o Git aparece e opera.
- Estas variáveis podem ser armazenadas em três lugares diferentes:
  - Git config do sistema como um todo.
  - Git config do usuário.
  - Git config do projeto.

Configurando o Git

- Sua Identidade

```
$ git config --global user.name "Fulano de Tal"
$ git config --global user.email fulanodetal@exemplo.br
```

Git CMD

```
C:\Users\USER>git config --global user.name "Pablo Vargas"
C:\Users\USER>git config --global user.email "pablo.vargas@unir.br"
```

Configurando o Git

- Testando suas configurações

```
$ git config --list
user.name=Fulano de Tal
user.email=fulanodetal@exemplo.br
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

Git CMD

```
C:\Users\USER>git config user.name
Pablo Vargas

C:\Users\USER>git config user.email
pablo.vargas@unir.br

C:\Users\USER>
```

Testando o Git

- Crie uma pasta na raiz chamada github e dentro dela outra pasta chamada Palomakoba
- Abra o terminal da IDE e digite os comandos abaixo

```
PS C:\github\Palomakoba> mkdir git_test


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        07/07/2022     10:11                git_test


PS C:\github\Palomakoba> cd git_test
PS C:\github\Palomakoba\git_test> git init
Initialized empty Git repository in C:/github/Palomakoba/git_test/.git/
PS C:\github\Palomakoba\git_test>
```

$ git init: inicializa o repositório para que o git reconheça.

Testando o Git

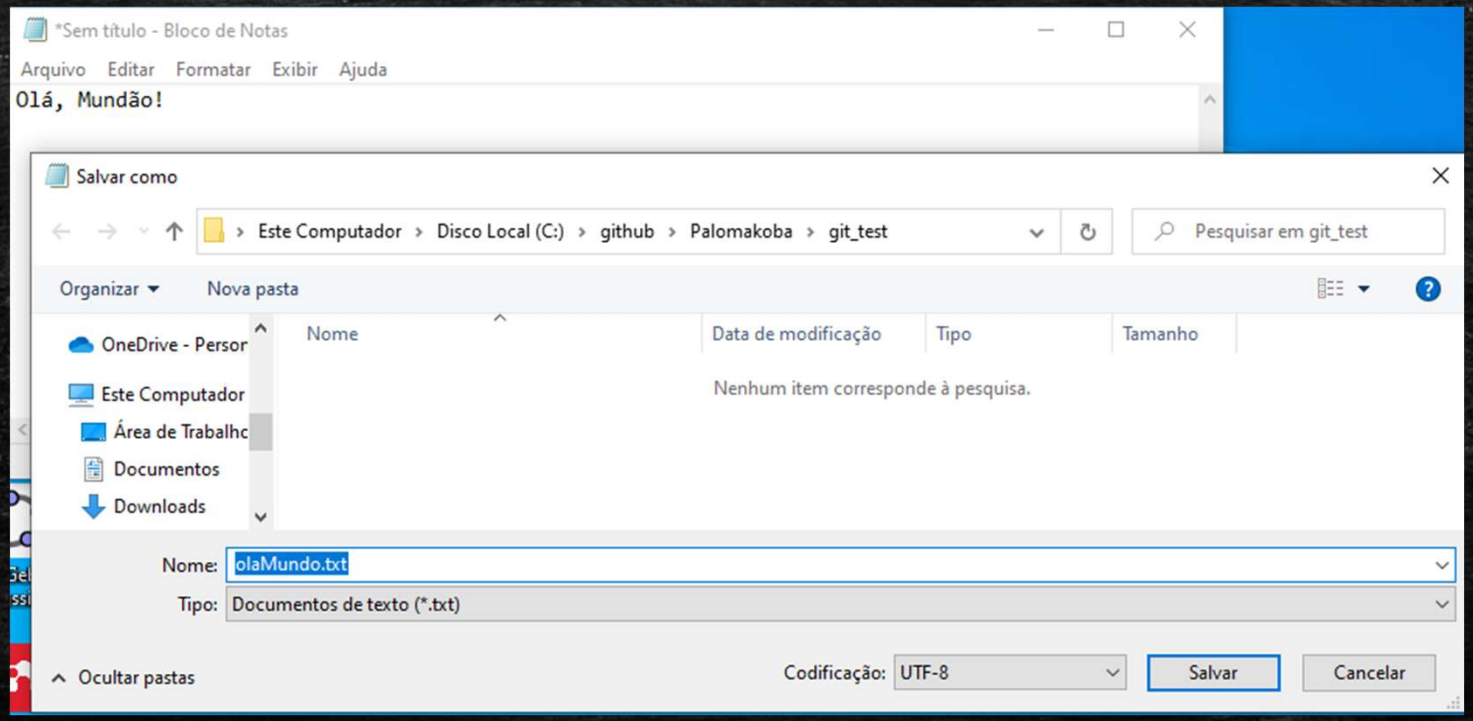
Git CMD

```
C:\github\Palomakoba\git_test>git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

C:\github\Palomakoba\git_test>
```

$ git status: é um comando para verificar quais arquivos estão em qual estado.

Testando o Git

- Crie um arquivo do Bloco de Notas com a seguinte frase: Olá, Mundão!
- Salve o arquivo na pasta C:\github\Palomakoba\git_test



Testando o Git

- Abra Git CMD na pasta git_test e adicione o comando abaixo

Git CMD

```
C:\github\Palomakoba\git_test>git add olaMundo.txt

C:\github\Palomakoba\git_test>
```

$ git add: é um comando que especifica o(s) arquivo(s) que você quer monitorar.

Testando o Git

Ao digitar "git status" verifique o que mudou as informações

Git CMD

```
C:\github\Palomakoba\git_test>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        olaMundo.txt

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

C:\github\Palomakoba\git_test>
```

Neste ponto você já tem um repositório Git mas o arquivo ainda não está monitorado e sem um commit inicial.

Testando o Git

- Ao digitar o comando $git add “nome do arquivo”
  - O arquivo começa a ser monitorado porém ainda sem um commit inicial.

Git CMD

```
C:\github\Palomakoba\git_test>git add olaMundo.txt

C:\github\Palomakoba\git_test>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   olaMundo.txt


C:\github\Palomakoba\git_test>_
```

Testando o Git

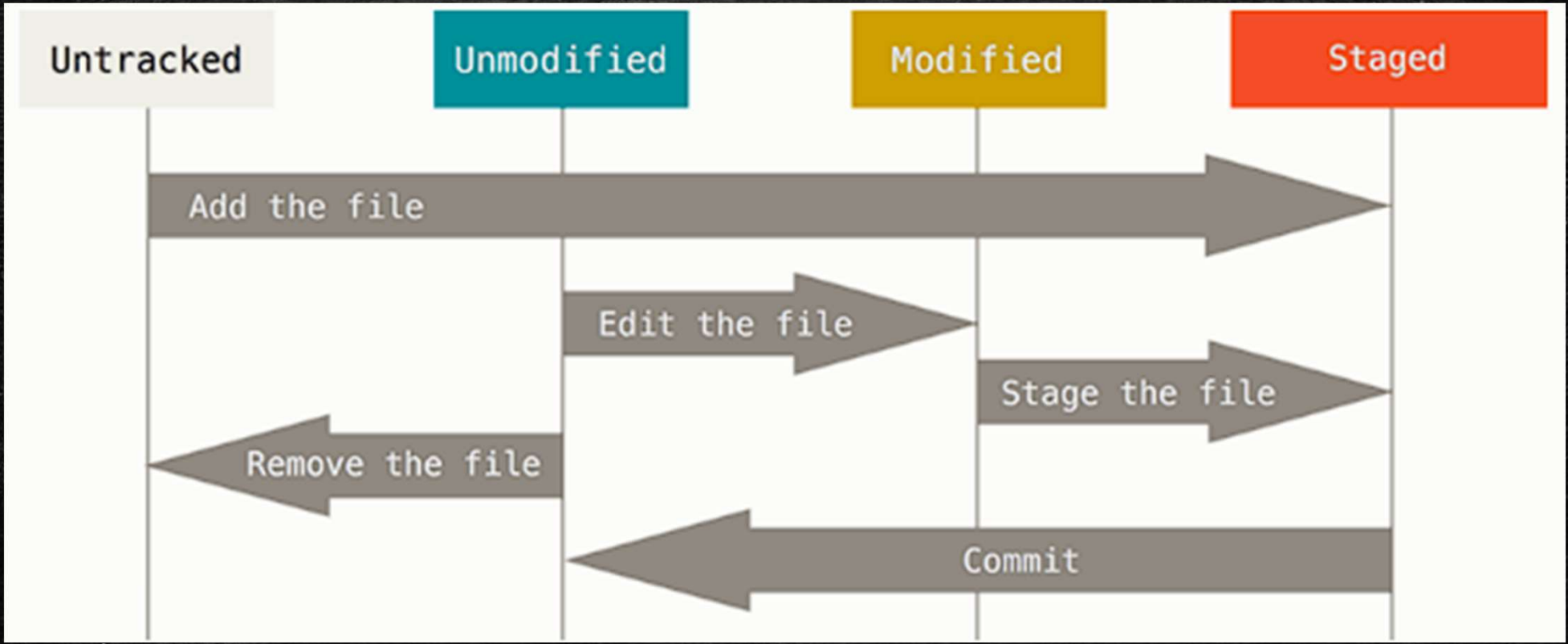
- O **commit** é um comando que pega todos os arquivos e cria uma imagem deles no repositório.
  - O –m serve para passar uma mensagem do que está sendo "commitado".

Git CMD

```
C:\github\Palomakoba\git_test>git commit -m "Arquivo para teste do git"
[master (root-commit) 116c348] Arquivo para teste do git
 1 file changed, 1 insertion(+)
 create mode 100644 olaMundo.txt

C:\github\Palomakoba\git_test>git status
On branch master
nothing to commit, working tree clean

C:\github\Palomakoba\git_test>
```

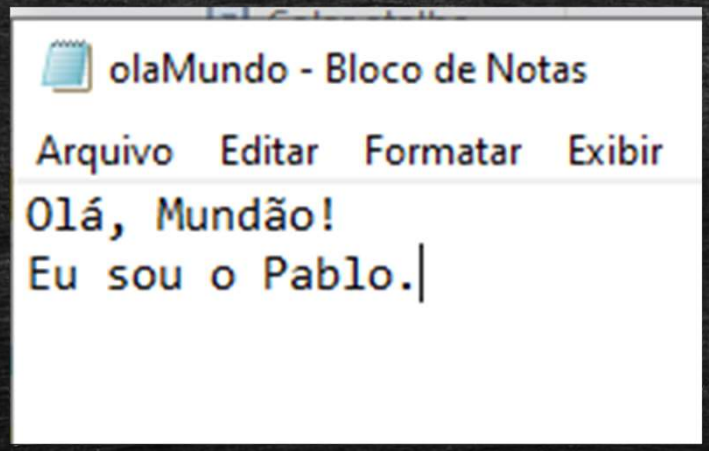
Veja que após commit foi realizado um git status informando que não tem mais nada para "commitar".

Ciclo de um arquivo no Git



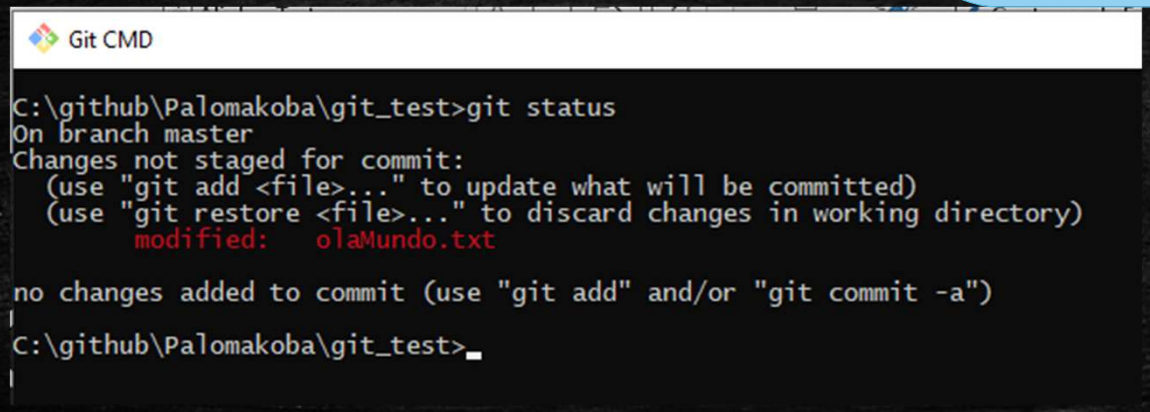
Ciclo de um arquivo no Git

Modifique o arquivo olaMundo.txt



Portanto, o arquivo quando editado passou para status de modificado (modified). Neste ponto, as alterações realizadas no arquivo ainda não foram "commitadas".

Digite o comando git status no Git CMD



Ciclo de um arquivo no Git

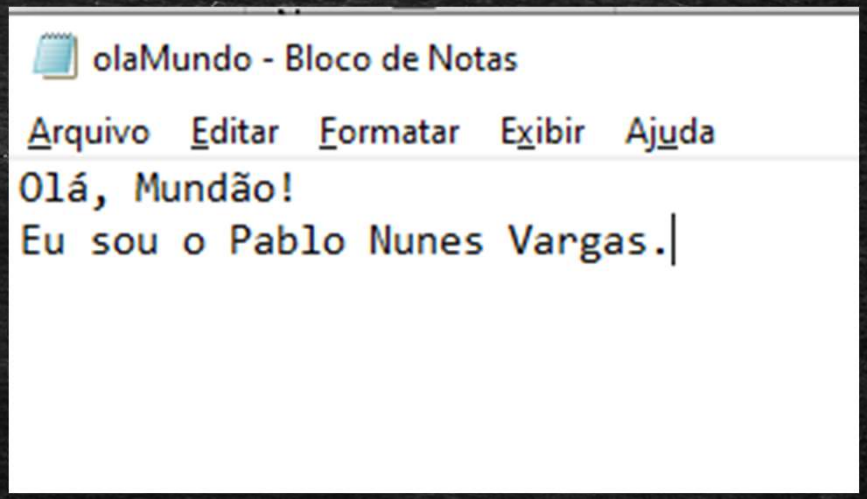
Git CMD

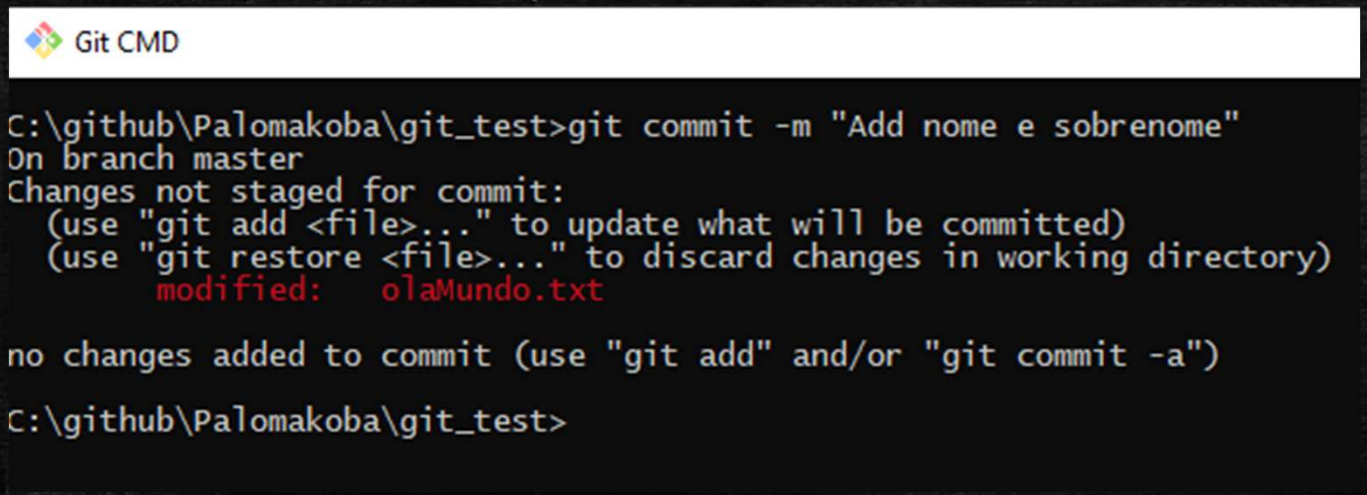
```
C:\github\Palomakoba\git_test>git add olaMundo.txt

C:\github\Palomakoba\git_test>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   olaMundo.txt


C:\github\Palomakoba\git_test>git commit -m "Add meu nome"
[master ea9cf69] Add meu nome
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Ao adicionar o arquivo novamente ele passa para o estado de staged, ou seja, quando ocorrer o próximo commit as informações alteradas vão ser gravadas.

Ciclo de um arquivo no Git

Modifique o arquivo olaMundo.txt novamente





Não é possível realizar o commit sem adicionar o arquivo”.

Ciclo de um arquivo no Git

Exercício: Adicione o arquivo olaMundo.txt e depois faça um commit com a mensagem “Add nome e sobrenome”

Fundamentos de Git

Git CMD

```
C:\github\Palomakoba\git_test>git log
commit 94a777e1747246f9a7d68b0b99fe5b5f14076947 (HEAD -> master)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 10:56:24 2022 -0400

    Add nome e sobrenome

commit ea9cf6908344589ca8f52a11617482c07f310e89
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 10:40:55 2022 -0400

    Add meu nome

commit 116c34840ad57d8a59f7a9bdd2fecb21c7b6f0f3
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 10:24:55 2022 -0400

    Arquivo para teste do git

C:\github\Palomakoba\git_test>
```

Comando $ git log serve para visualizar o histórico de commits

Git CMD

```
C:\github\Palomakoba\git_test>git log --author="Pablo"
commit 94a777e1747246f9a7d68b0b99fe5b5f14076947 (HEAD -> master)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 10:56:24 2022 -0400

    Add nome e sobrenome

commit ea9cf6908344589ca8f52a11617482c07f310e89
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 10:40:55 2022 -0400

    Add meu nome

commit 116c34840ad57d8a59f7a9bdd2fecb21c7b6f0f3
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 10:24:55 2022 -0400

    Arquivo para teste do git

C:\github\Palomakoba\git_test>
```

Comando $ git log --author="Fulano" serve para visualizar o histórico de commits de um autor específico.

Fundamentos de Git

Git CMD

```
C:\github\Palomakoba\git_test>git shortlog -sn
     3  Pablo Vargas

C:\github\Palomakoba\git_test>git log --graph
* commit 94a777e1747246f9a7d68b0b99fe5b5f14076947 (HEAD -> master)
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Fri Jul 8 10:56:24 2022 -0400
|
|     Add nome e sobrenome
|
* commit ea9cf6908344589ca8f52a11617482c07f310e89
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Fri Jul 8 10:40:55 2022 -0400
|
|     Add meu nome
|
* commit 116c34840ad57d8a59f7a9bdd2fecb21c7b6f0f3
  Author: Pablo Vargas <pablo.vargas@unir.br>
  Date:   Fri Jul 8 10:24:55 2022 -0400

      Arquivo para teste do git

C:\github\Palomakoba\git_test>
```

Comando $ git log --graph serve para visualizar o histórico de commits de forma gráfica. Muito útil para visualizar os branch, merge e rebase.

Fundamentos de Git

Git CMD

```
C:\github\Palomakoba\git_test>git shortlog
Pablo Vargas (3):
      Arquivo para teste do git
      Add meu nome
      Add nome e sobrenome


C:\github\Palomakoba\git_test>git shortlog -sn
     3  Pablo Vargas

C:\github\Palomakoba\git_test>_
```

Comando $ git shortlog serve para visualizar o histórico de commits de forma resumida.

Fundamentos de Git

Git CMD

```
C:\github\Palomakoba\git_test>git log --graph
* commit 94a777e1747246f9a7d68b0b99fe5b5f14076947 (HEAD -> master)
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Fri Jul 8 10:56:24 2022 -0400
|
|     Add nome e sobrenome
|
* commit ea9cf6908344589ca8f52a11617482c07f310e89
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Fri Jul 8 10:40:55 2022 -0400
|
|     Add meu nome
|
* commit 116c34840ad57d8a59f7a9bdd2fecb21c7b6f0f3
  Author: Pablo Vargas <pablo.vargas@unir.br>
  Date:   Fri Jul 8 10:24:55 2022 -0400

      Arquivo para teste do git

C:\github\Palomakoba\git_test>git show ea9cf6908344589ca8f52a11617482c07f310e89
commit ea9cf6908344589ca8f52a11617482c07f310e89
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 10:40:55 2022 -0400

    Add meu nome

diff --git a/olaMundo.txt b/olaMundo.txt
index bd1f80f..1a7c99e 100644
--- a/olaMundo.txt
+++ b/olaMundo.txt
@@ -1 +1,2 @@
-Olá, Mundão!
\ No newline at end of file
+Olá, Mundão!
+Eu sou o Pablo.
\ No newline at end of file

C:\github\Palomakoba\git_test>
```

Comando $ git show serve para visualizar as modificações que ocorreram em determinado commit e outras informações.

Fundamentos de Git

Modifique o arquivo olaMundo.txt novamente

Bom para revisar antes de enviar para um commit

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Vamos aprender os fundamentos do git?
```

Comando $ git diff mostra as diferenças do arquivo Head para o commit anterior.

Git CMD

```
C:\github\Palomakoba\git_test>git diff
diff --git a/olaMundo.txt b/olaMundo.txt
index 2d8f0ad..e8c271a 100644
--- a/olaMundo.txt
+++ b/olaMundo.txt
@@ -1,2 +1,3 @@
 Olá, Mundão!
-Eu sou o Pablo Nunes Vargas.
\ No newline at end of file
+Eu sou o Pablo Nunes Vargas.
+Vamos aprender os fundamentos do git?
\ No newline at end of file

C:\github\Palomakoba\git_test>git diff --name-only
olaMundo.txt

C:\github\Palomakoba\git_test>
```

Fundamentos de Git

Git CMD

```
C:\github\Palomakoba\git_test>git commit -am "Adicionei uma pergunta"
[master d16e532] Adicionei uma pergunta
 1 file changed, 2 insertions(+), 1 deletion(-)

C:\github\Palomakoba\git_test>_
```

$ git commit -am “msg” é um comando útil para quando o arquivo já pertencia a um commit e foi apenas editado.

Não necessita adicionar o arquivo para staged ($ git add olaMundo.txt)

Fundamentos de Git

Modifique o arquivo olaMundo.txt novamente

Com o comando $ git checkout nomeDoArquivo é possível reverter a ultima edição realizada

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Vamos aprender os fundamentos do git?
Sim
Site referencia para aprender: https://git-scm.com/book/pt-br/v2
```

Git CMD

```
C:\github\Palomakoba\git_test>git checkout olaMundo.txt
Updated 1 path from the index

C:\github\Palomakoba\git_test>git diff

C:\github\Palomakoba\git_test>
```

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Vamos aprender os fundamentos do git?
```

Fundamentos de Git

Modifique o arquivo olaMundo.txt novamente

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Vamos aprender os fundamentos do git?
Claro!
```

Mas e se caso tenha sido adicionado para o staged. Utilizar o $ git reset HEAD nomeDoArquivo.

Git CMD

```
C:\github\Palomakoba\git_test>git diff
diff --git a/olaMundo.txt b/olaMundo.txt
index e8c271a..e55a7b3 100644
--- a/olaMundo.txt
+++ b/olaMundo.txt
@@ -1,3 +1,4 @@
 Olá, Mundão!
 Eu sou o Pablo Nunes Vargas.
-Vamos aprender os fundamentos do git?
\ No newline at end of file
+Vamos aprender os fundamentos do git?
+Claro!

C:\github\Palomakoba\git_test>git add olaMundo.txt

C:\github\Palomakoba\git_test>gis status
'gis' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

C:\github\Palomakoba\git_test>git reset HEAD olaMundo.txt
Unstaged changes after reset:
M       olaMundo.txt

C:\github\Palomakoba\git_test>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   olaMundo.txt

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

C:\github\Palomakoba\git_test>
```

Fundamentos de Git

Modifique o arquivo olaMundo.txt novamente

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Vamos aprender os fundamentos do git?
Claro!
```

Mas e se caso tenha sido "commitado" . Utilizar o $ git reset --soft/mixed/hard "hash do commit anterior ao que foi feito".

```
C:\github\Palomakoba\git_test>git log
commit 546d5f8ccd2c618f45685ce44e6e81dcd6c73b46 (HEAD -> master)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 14:53:48 2022 -0400

    Resposta Claro

commit d16e532236a3007abc6294929fa8d3eb2d5b58e9
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 12:15:3[illegible] 2022 -0400

    Adicionei uma pergunta

commit 94a777e1747246f9a7[illegible]68b0b99fe5b5f14076947
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Fri Jul 8 10:56:2[illegible] 2022 -0400
```

```
C:\github\Palomakoba\git_test>git reset --soft d16e532236a3007abc6294929fa8d3eb2d5b58e9

C:\github\Palomakoba\git_test>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   olaMundo.txt
```

soft: volta o arquivo modificado na staged

mixed: volta o arquivo modificado antes da staged

hard: volta sem o arquivo modificado

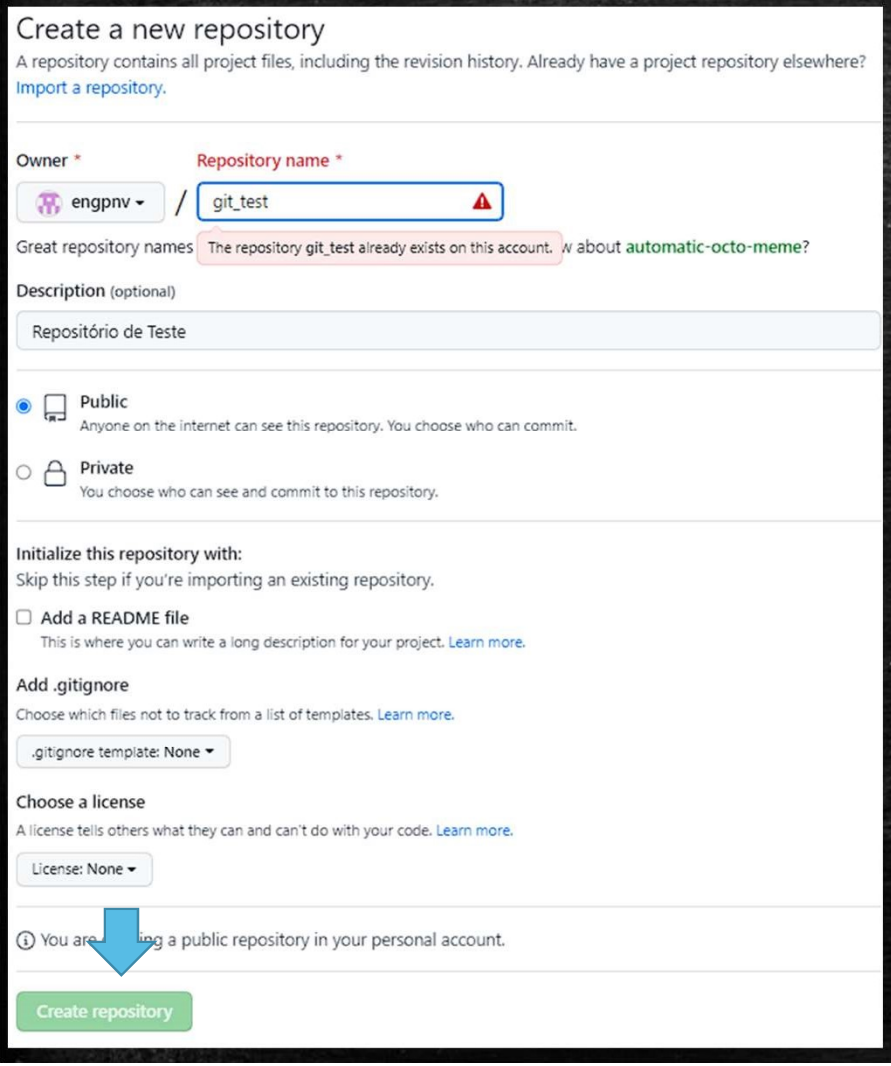
Clonando um Repositório Existente

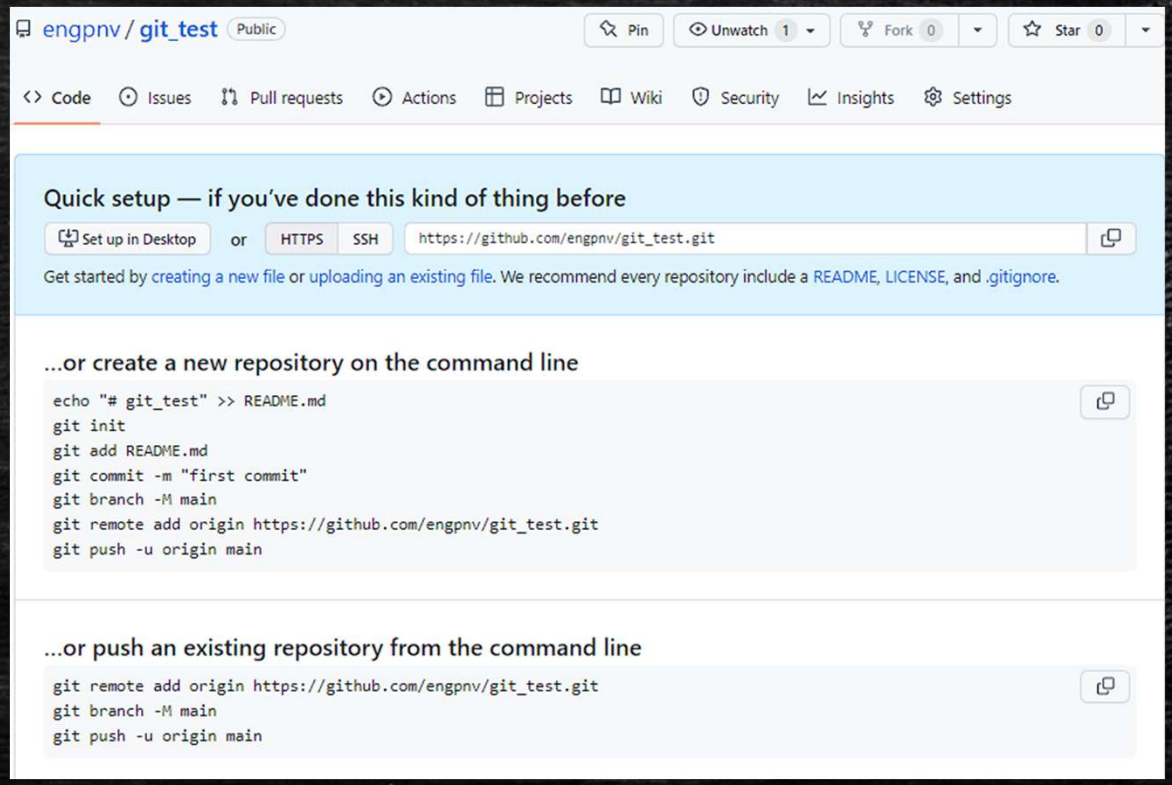
- Caso você queira obter a **cópia de um repositório** Git existente (por exemplo, um projeto que você queira contribuir), o comando para isso é **git clone**.

```
C:\github\Palomakoba\git_test>cd ..

C:\github\Palomakoba>git clone https://github.com/libgit2/libgit2 Meu_libgit
Cloning into 'Meu_libgit'...
remote: Enumerating objects: 117602, done.
remote: Counting objects: 100% (117602/117602), done.
remote: Compressing objects: 100% (32258/32258), done.
remote: Total 117602 (delta 83432), reused 117529 (delta 83365), pack-reused 0
Receiving objects: 100% (117602/117602), 59.45 MiB | 10.88 MiB/s, done.

Resolving deltas: 100% (83432/83432), done.
Updating files: 100% (11500/11500), done.

C:\github\Palomakoba>
```

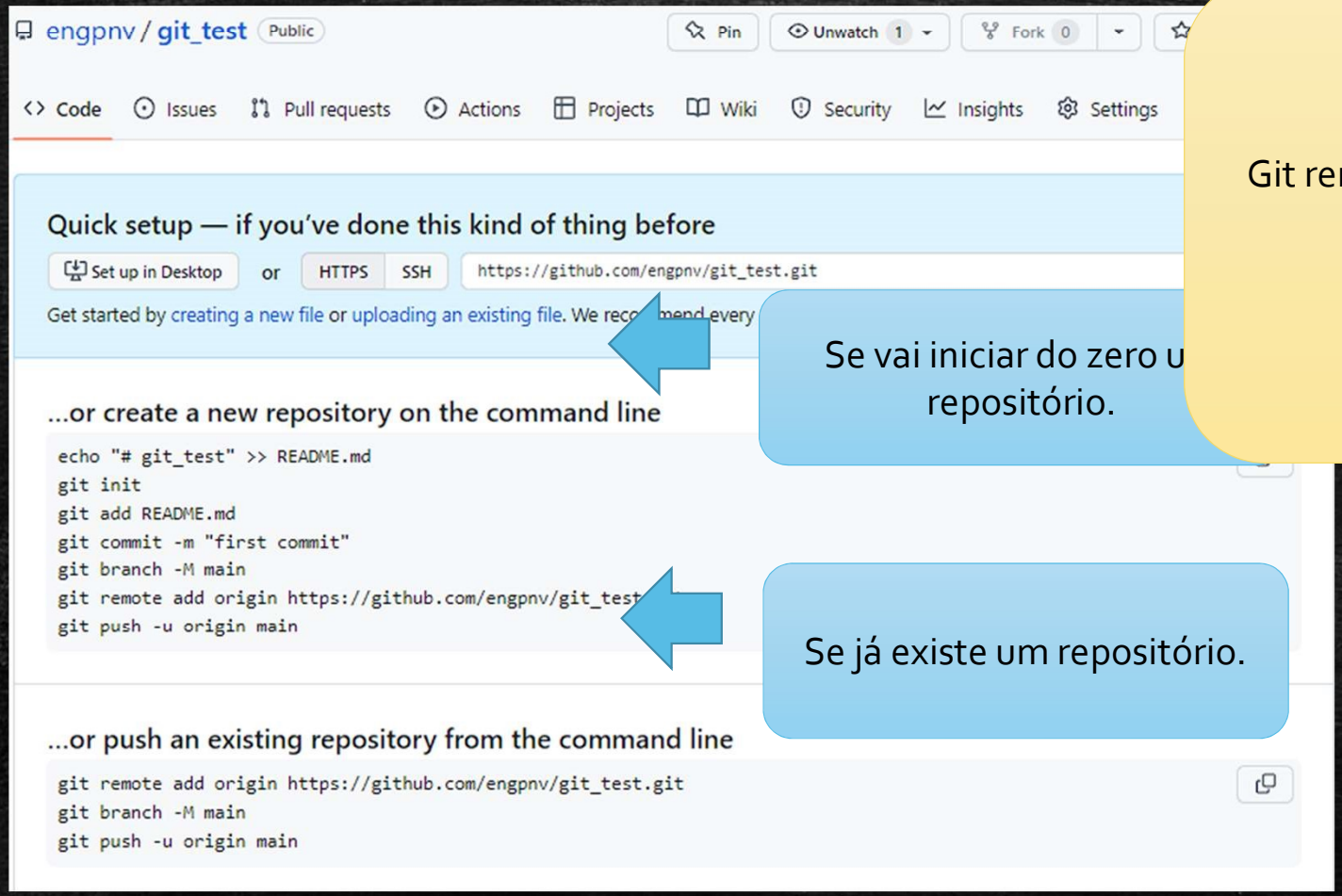
Criando um Repositório Remoto

Preencha os dados do repositório conforme imagem ao lado.





Criando um Repositório Remoto



Se vai iniciar do zero u
repositório.

Se já existe um repositório.

Git remote add origin "linkRepositorio"
Git branch –M main
Git push –u origin main

Criando um Repositório Remoto

SSH

É um **protocolo** de rede que permite que a conexão com determinados servidores por meio de uma comunicação **criptografada,** trazendo mais **segurança** para as transações de dados.

Gera uma chave SSH: https://docs.github.com/pt/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

Adiciona uma chave SSH: https://docs.github.com/pt/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

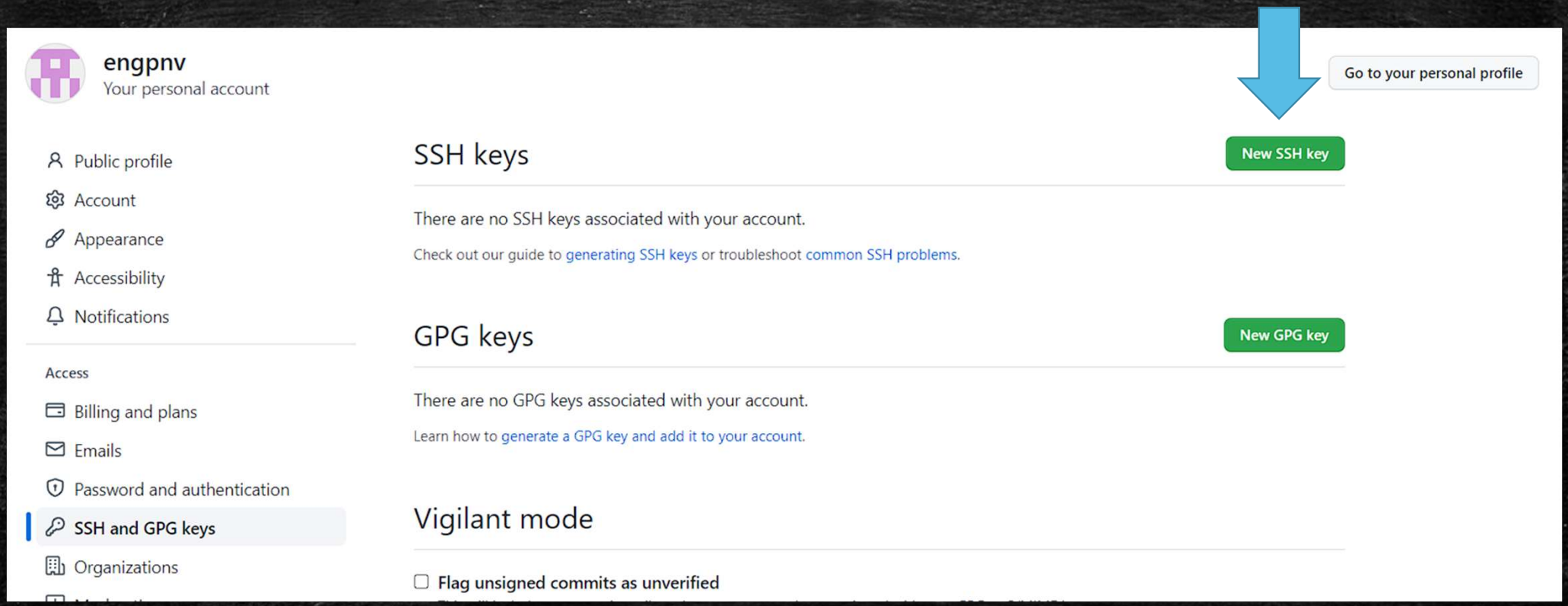
Criando um Repositório Remoto

```
C:\github\Palomakoba\git_test>ssh-keygen -t ed25519 -C "pablo.vargas@unir.br"
Generating public/private ed25519 key pair.
Enter file in which to save the key (C:\Users\USER/.ssh/id_ed25519):
C:\Users\USER/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\USER/.ssh/id_ed25519.
Your public key has been saved in C:\Users\USER/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:QIWa8jeZrenAZgWKIBVeQunfO4gwYWTHHjLHulVjsoc pablo.vargas@unir.br
The key's randomart image is:
+--[ED25519 256]--+
| .Bo. .o.        |
| B.X =.          |
|=.O Ooo          |
|++oEoo .         |
|o.++...+S        |
|o. .o.* .        |
| o .=o =         |
|  .o..=          |
|     ...         |
+----[SHA256]-----+
```

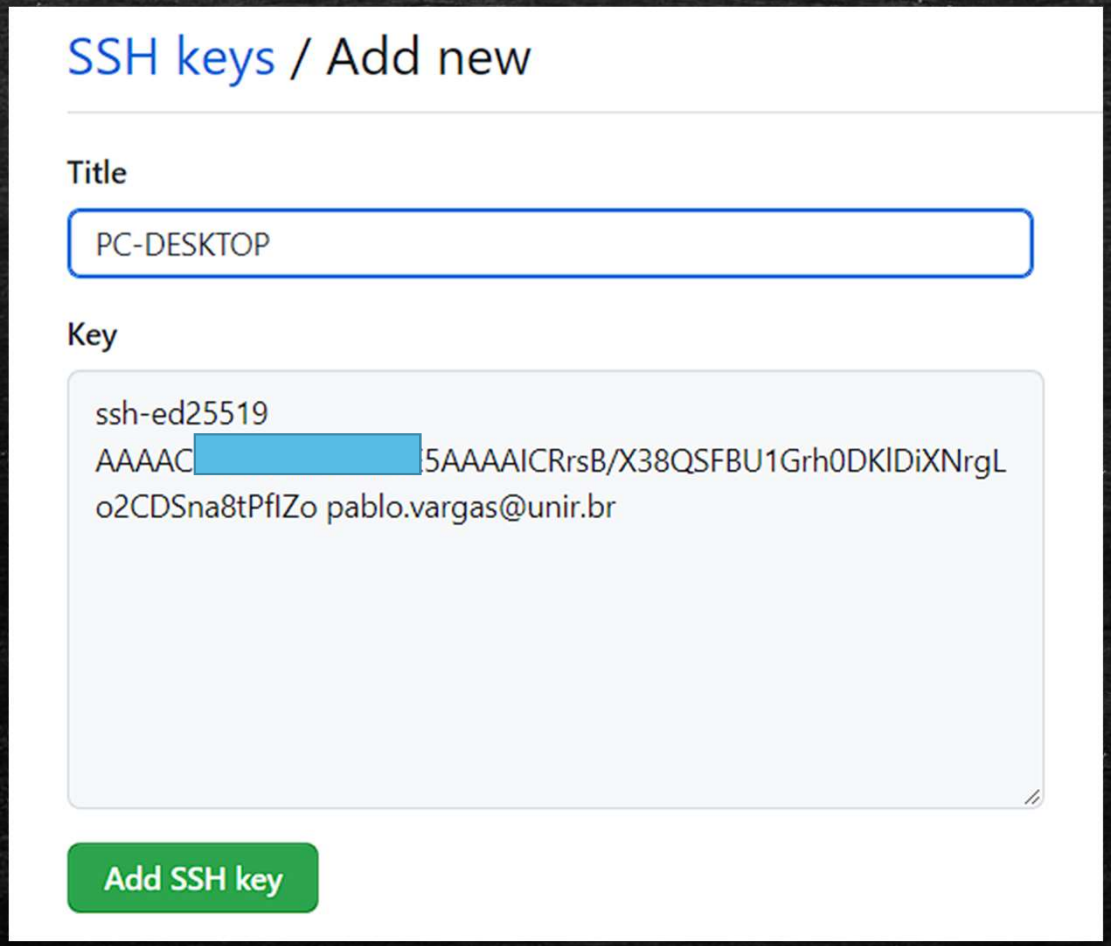
```
C:\Users\USER\.ssh>dir
 O volume na unidade C não tem nome.
 O Número de Série do Volume é 7C0F-DC6A

 Pasta de C:\Users\USER\.ssh

08/07/2022  15:41    <DIR>          .
08/07/2022  15:41    <DIR>          ..
08/07/2022  15:42               411 id_ed25519
08/07/2022  15:42               103 id_ed25519.pub
               2 arquivo(s)            514 bytes
               2 pasta(s)   242.264.907.776 bytes disponíveis

C:\Users\USER\.ssh>cat id_ed25519.pub
'cat' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

C:\Users\USER\.ssh>more id_ed25519.pub
ssh-ed25519 AAAAC[redacted]AAAAICRrsB/X38QSFBU1GrhODKlDiXNrgLo2CDSna8tPfIZo pablo.vargas@unir.br

C:\Users\USER\.ssh>
```

Copie a chave

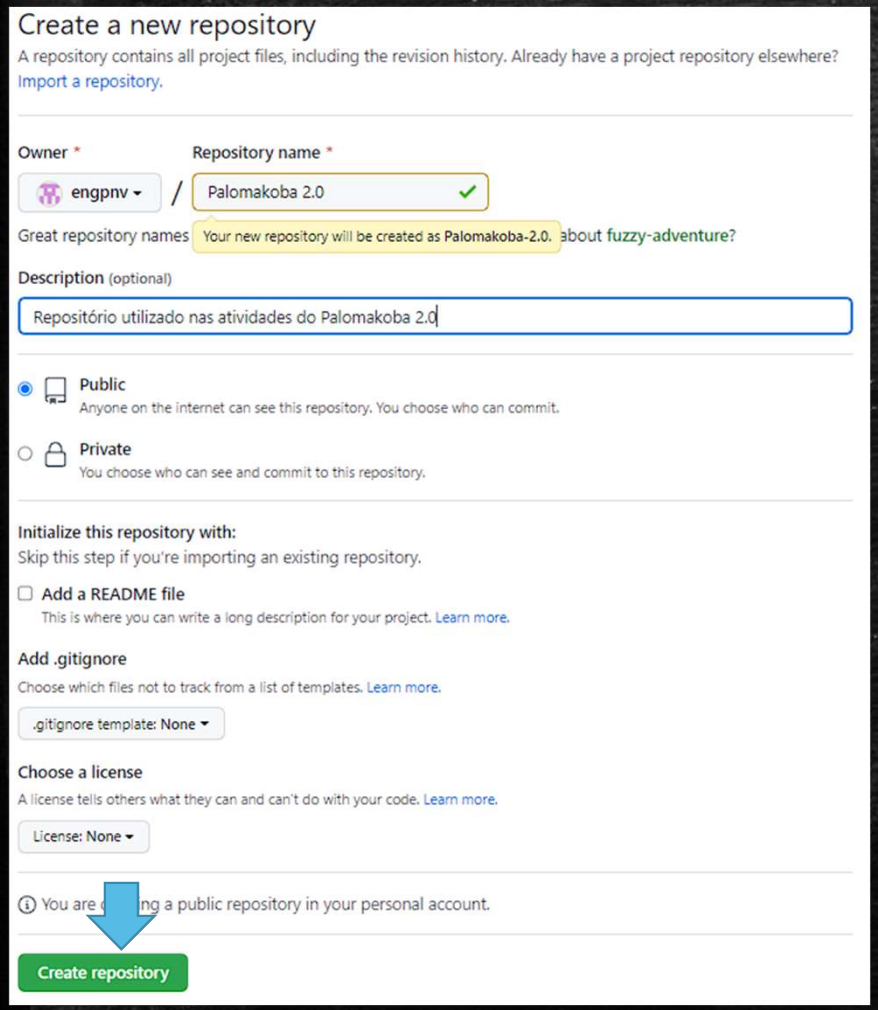
Criando um Repositório Remoto



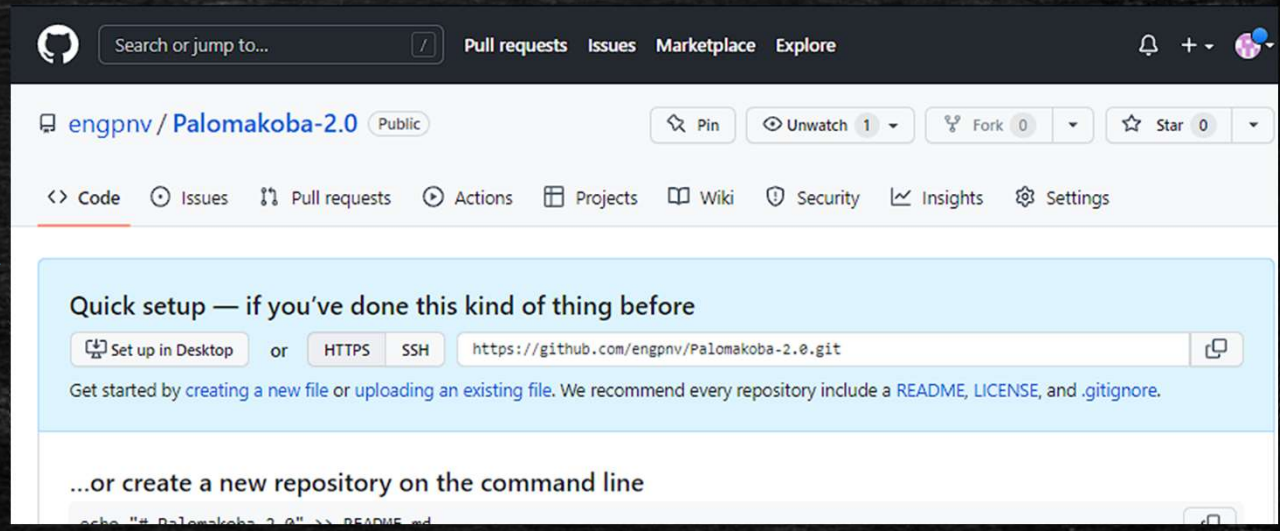
Criando um Repositório Remoto



Criando um Repositório Remoto

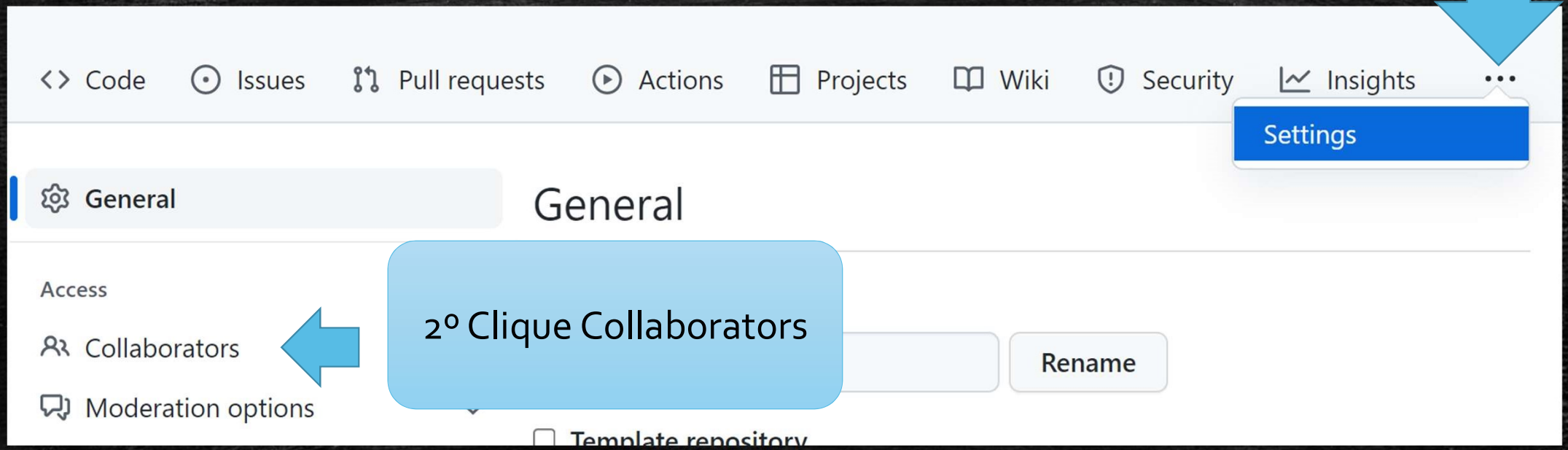


Preencha os dados do repositório conforme imagem ao lado.



Liberando acesso

1º Clique ...>Settings



2º Clique Collaborators

Manage access

Adicione sua equipe

You haven't invited any collaborators yet

Add people

Criando um Repositório Remoto

Exercício:

- PO cria um bloco de notas no repositório Palomakoba
- SM cria os atributos conforme imagem ao lado
- Demais membros insere os seus dados

Git remote add origin "linkRepositorio"
Git branch –M main
Git push –u origin main

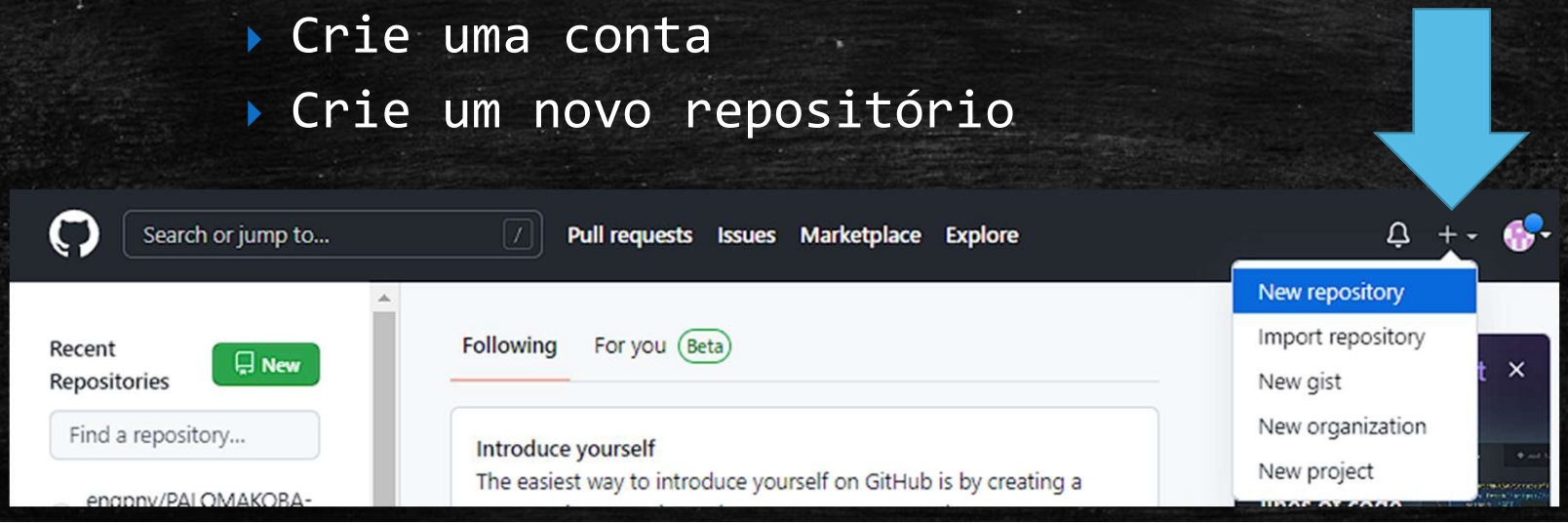
infoEquipe - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
(PO)
Product Owner: Fulano de Tal
Equipe: A
(SM)
Scrum Mascter: Ciclana\ de Tal

        Desenvolvedor 1: (Devs) Beltrano de Tal
        Habilidades:
        Experiência:

        Desenvolvedor 2: (Devs)
        Habilidades:
        Experiência:
```

Criando um Repositório Remoto

- **Repositório remoto**: é um serviço na nuvem no qual você salva o projeto podendo ser público ou não, funcionando como uma rede social.
  - Ex: Github, Gitlab e Bitbucket...
  - Abra o site do github (https://github.com/)
    - Crie uma conta
    - Crie um novo repositório



Enviando Mudanças para o remoto

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/engpnv/git_test.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# git_test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/engpnv/git_test.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/engpnv/git_test.git
git branch -M main
git push -u origin main
```

Git CMD

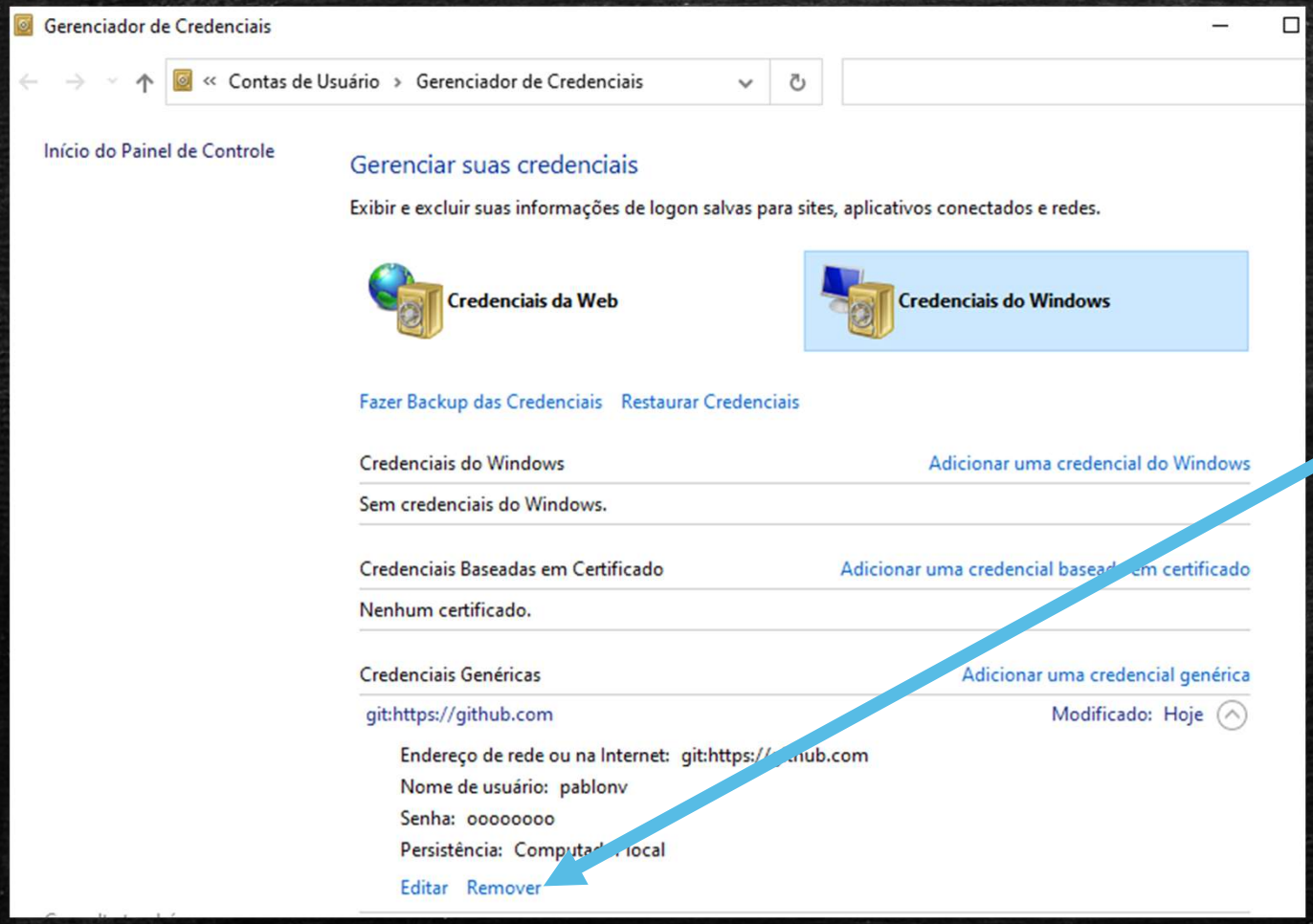
```
C:\Users\USER>cd ..
C:\Users>cd ..
C:\>cd github
C:\github>cd Palomakoba
C:\github\Palomakoba>cd git_test
C:\github\Palomakoba\git_test>git remote add origin https://github.com/engpnv/git_test.git
error: remote origin already exists.
C:\github\Palomakoba\git_test>git remote add teste https://github.com/engpnv/git_test.git
C:\github\Palomakoba\git_test>git branch -M main
C:\gith lomakoba\git_test>git push -u origin main
```

Enviando Mudanças para o remoto

```
C:\github\Palomakoba\git_test>git remote add teste https://

C:\github\Palomakoba\git_test>git branch -M main

C:\github\Palomakoba\git_test>git push -u origin main
remote: Permission to engpnv/git_test.git denied to pablonv
fatal: unable to access 'https://github.com/engpnv/git_test                    urned error: 403

C:\github\Palomakoba\git_test>
```

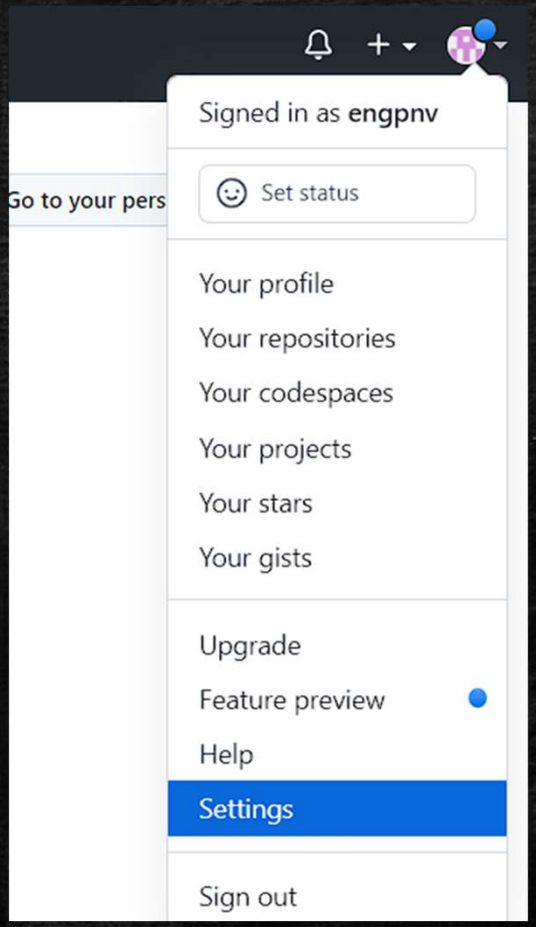
Se houver problema de permissão então será necessário realizar uma modificação no Gerenciador de Credenciais

Enviando Mudanças para o remoto

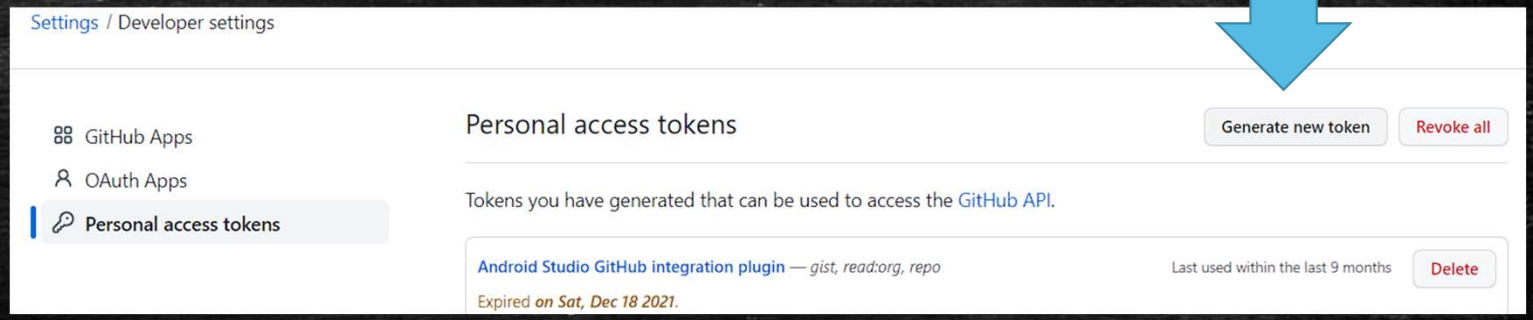


Clique em remover as credenciais do git

Enviando Mudanças para o remoto

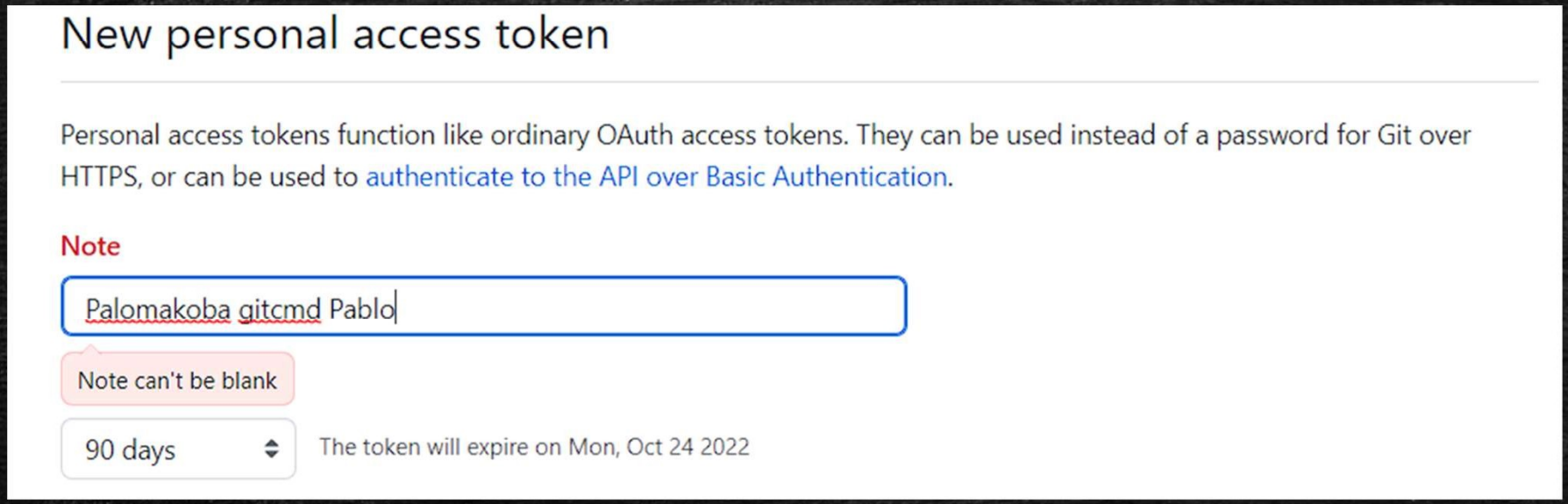


Clique em Settings >> Developer Setting



Clique para gerar um novo token

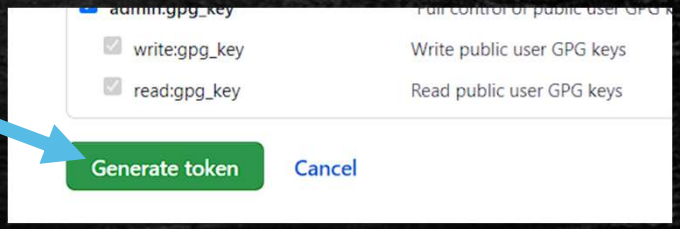
Enviando Mudanças para o remoto



Escrava uma nota e o tempo de duração do token

Selecione todas as opções de escopo

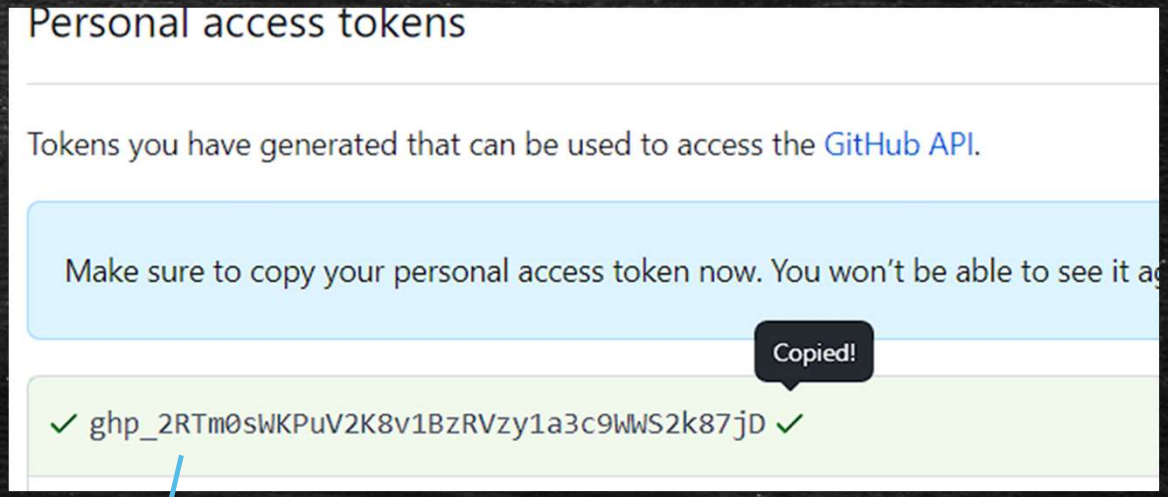
Clique em Generate token

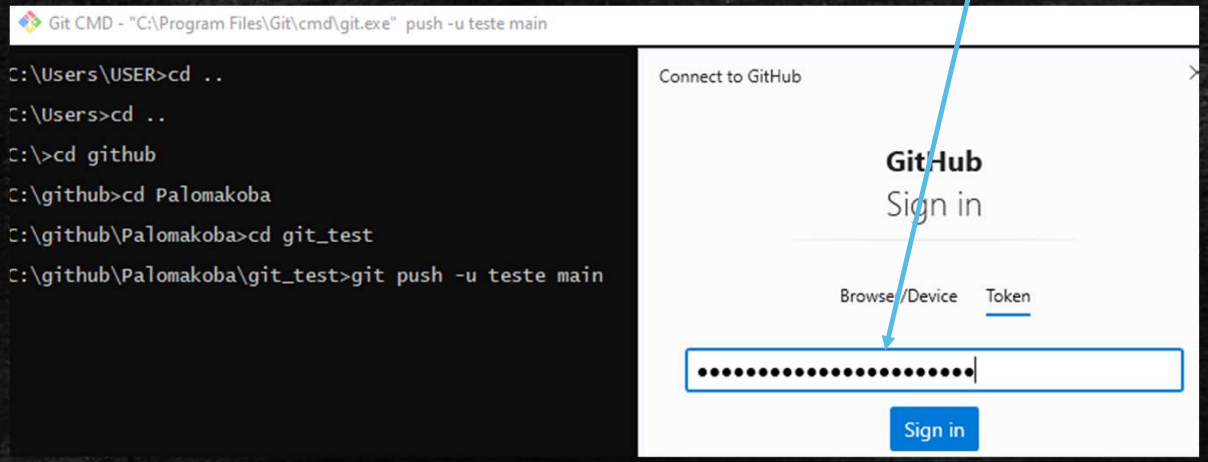


Enviando Mudanças para o remoto

Copie a chave gerada e refaça a operação de push no git CMD

Na nova janela: vá em Token, cole a chave e depois clique em "Sign in"

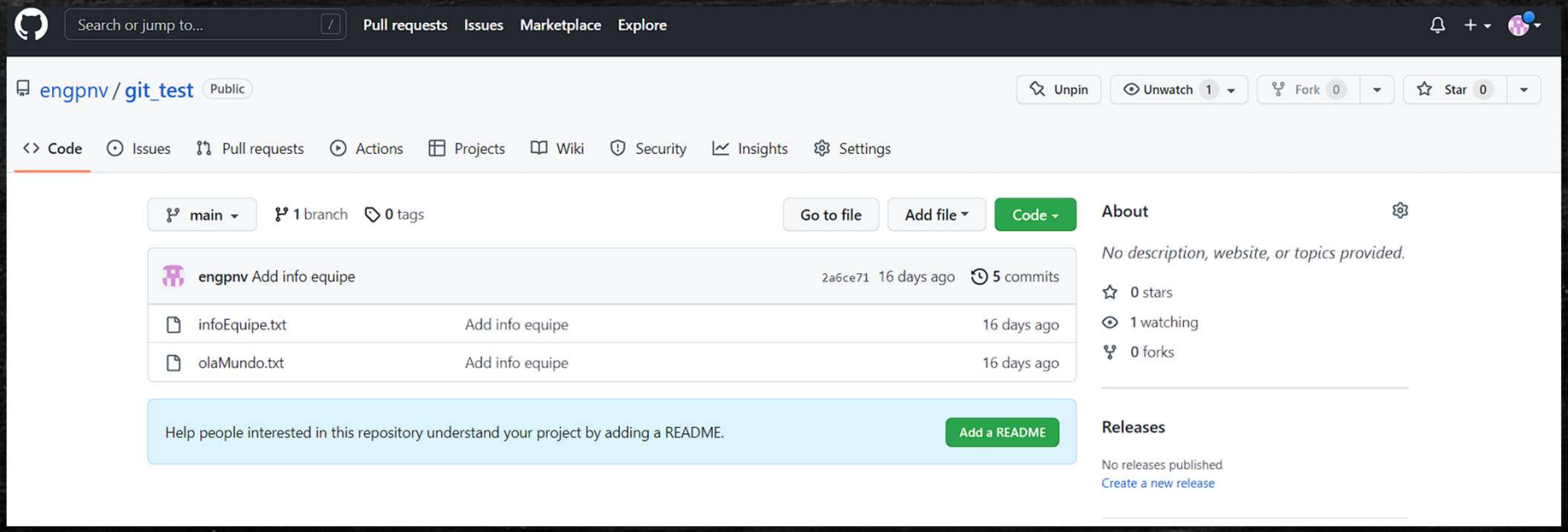





```
C:\github\Palomakoba\git_test>git push -u teste main
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (16/16), 1.43 KiB | 1.43 MiB/s, done.
Total 16 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/engpnv/git_test.git
 * [new branch]      main -> main
branch 'main' set up to track 'teste/main'.

C:\github\Palomakoba\git_test>
```

Enviando Mudanças para o remoto

Verifique se os arquivos foram enviando para o repositório remoto



Enviando Mudanças para o remoto

Edite e salve o arquivo olaMundo

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Vamos aprender os fundamentos do git?
Claro!
Qual é seu nome aluno?
Fulano de Tal
```

O comando de status informa que existe modificações que ainda não foram "commitadas"

```
C:\github\Palomakoba\git_test>git status
On branch main
Your branch is up to date with 'teste/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   olaMundo.txt

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

C:\github\Palomakoba\git_test>git commit -am "Add nome do aluno"
[main 53e2097] Add nome do aluno
 1 file changed, 2 insertions(+)

C:\github\Palomakoba\git_test>
C:\github\Palomakoba\git_test>
```

Faça o comando de commit adicionando uma mensagem referente a modificação

Enviando Mudanças para o remoto

```
C:\github\Palomakoba\git_test>git log
commit 53e2097e4af59a08541eae9dcd3fe22bf77fc0d6 (HEAD -> main)
Author: pablonv <pablo.vargas@unir.br>
Date:   Tue Jul 26 12:00:59 2022 -0400

    Add nome do aluno
```

Entretanto esse novo commit não está no remoto apesar de já aparecer no local

main git_test / olaMundo.txt

engpnv Add info equipe

1 contributor

4 lines (4 sloc) | 89 Bytes

```
1  Olá, Mundão!
2  Eu sou o Pablo Nunes Vargas.
3  Vamos aprender os fundamentos do git?
4  Claro!
```

engpnv / git_test Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

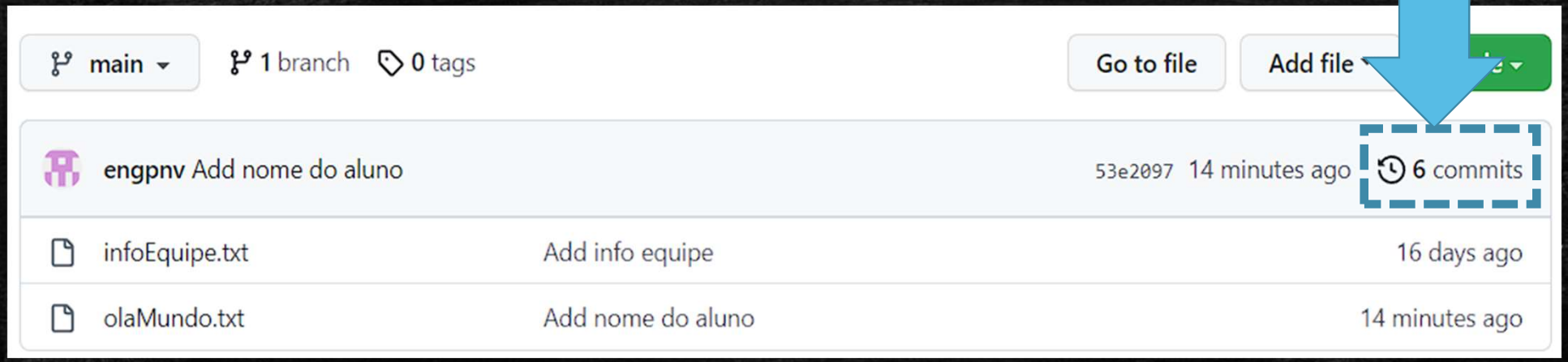
engpnv Add info equipe 2a6ce71 16 days ago 5 commits

infoEquipe.txt	Add info equipe	16 days ago
olaMundo.txt	Add info equipe	16 days ago

Enviando Mudanças para o remoto

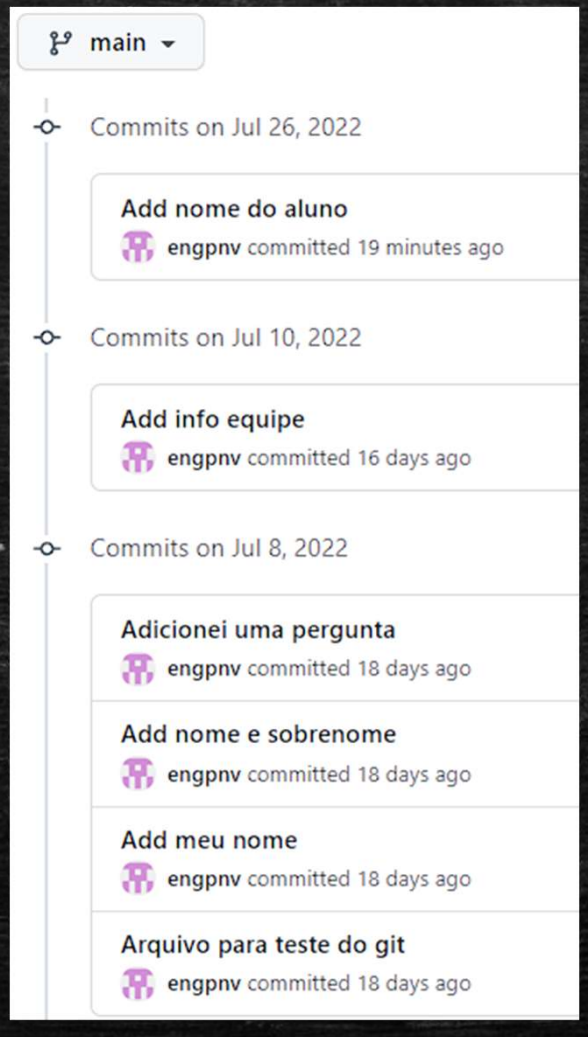
Utilize o comando git push origin main e então será enviando para o remoto as modificações adicionando um novo branch

```
C:\github\Palomakoba\git_test>git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 346 bytes | 346.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/engpnv/git_test.git
   2a6ce71..53e2097  main -> main

C:\github\Palomakoba\git_test>
```



Enviando Mudanças para o remoto

Podemos verificar todos os commits realizados e assim ter um maior controle sobre as versões dos arquivos



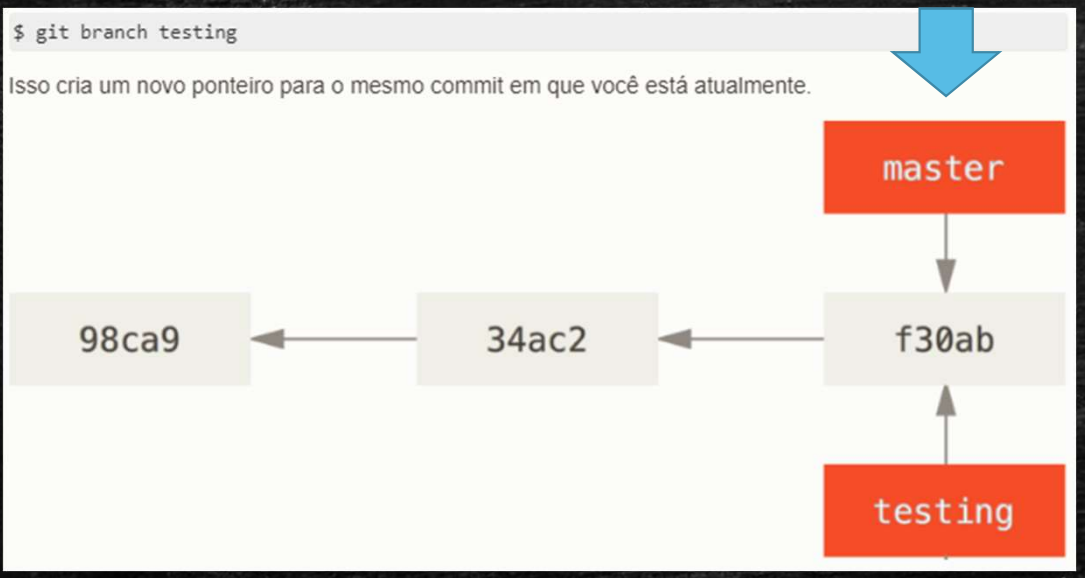
Fazendo branch no git

Branch: é um ponteiro móvel que leva a um commit.
+ Pode modificar sem alterar o local principal (main)
+ Facilmente "desligável"
+ Multiplas pessoas trabalhando
+ Evita conflitos

HEAD

```
$ git branch testing
```

Isso cria um novo ponteiro para o mesmo commit em que você está atualmente.




```
C:\github\Palomakoba\git_test>git branch testando

C:\github\Palomakoba\git_test>git log
commit 53e2097e4af59a08541eae9dcd3fe22bf7[illegible]fc0d6 (
HEAD -> main, origin/main, origin/HEAD, testando)

Author: pablonv <pablo.vargas@unir.br>
Date:   Tue Jul 26 12:00:59 2022 -0400

    Add nome do aluno
```

Fazendo branch no git

Git CMD - "C:\Program Files\Git\cmd\git.exe" log

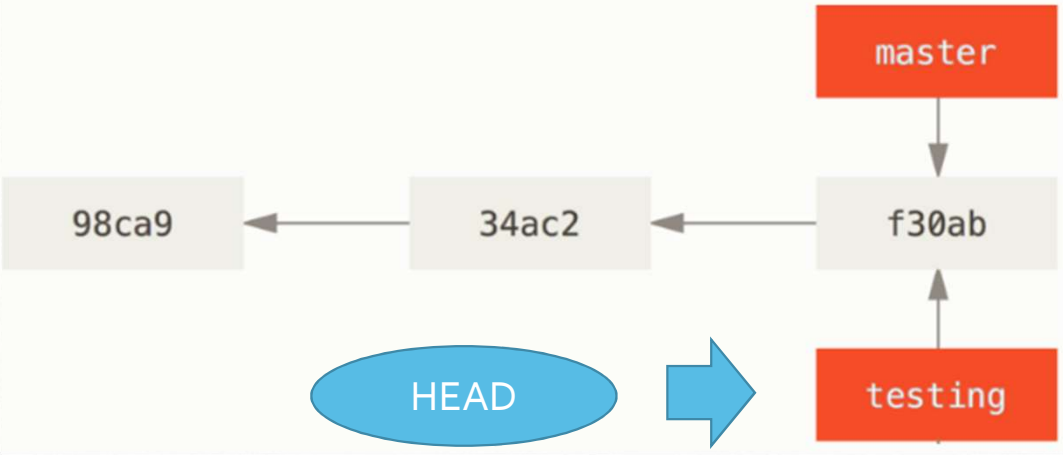
```
C:\github\Palomakoba\git_test>git checkout testando
Switched to branch 'testando'

C:\github\Palomakoba\git_test>git log
commit 53e2097e4af59a08541eae9dcd3fe22bf77fc0d6 (HEAD -> testando, origin/main, origin/HEAD, main)
Author: pablonv <pablo.vargas@unir.br>
Date:   Tue Jul 26 12:00:59 2022 -0400

    Add nome do aluno
```

HEAD: um ponteiro especial que serve para indicar para o git em qual branch está atualmente.

Comando git checkout "nomeDoBranch" serve para mudar o HEAD a um determinado Branch

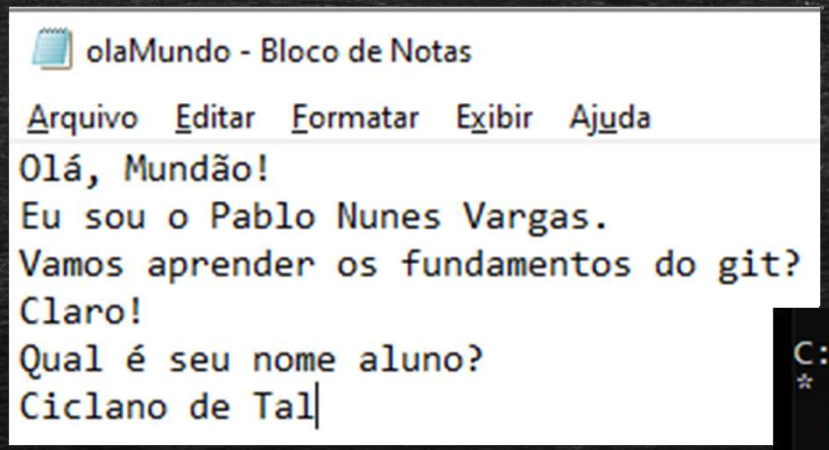
```
$ git branch testing
```

Isso cria um novo ponteiro para o mesmo commit em que você está atualmente.



Fazendo branch no git

Altere o arquivo do lado conforme imagem



Comando "git branch" serve para mostrar as branches existentes

```
C:\github\Palomakoba\git_test>git branch
* main
  testando
```

Faça um commit no Branch "testando" e verifique o log

```
C:\github\Palomakoba\git_test>git status
On branch testando
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   olaMundo.txt

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

C:\github\Palomakoba\git_test>git commit -am "Modificou nome do aluno"
[testando 3a50844] Modificou nome do aluno
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\github\Palomakoba\git_test>git status
On branch testando
nothing to commit, working tree clean
```

Fazendo branch no git

Git CMD - "C:\Program Files\Git\cmd\git.exe" log

```
C:\github\Palomakoba\git_test>git log
commit 3a508441080347aebad946f05088dab69e0623d3 (HEAD -> testando)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Wed Jul 27 20:50:57 2022 -0400

    Modificou nome do aluno

commit 53e2097e4af59a08541eae9dcd3fe22bf77fc0d6 (origin/main, origin/HEAD, main)
Author: pablonv <pablo.vargas@unir.br>
Date:   Tue Jul 26 12:00:59 2022 -0400

    Add nome do aluno

commit 2a6ce717d7e57806cbc873b22db3b200afad526d
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Sun Jul 10 09:37:56 2022 -0400

    Add info equipe
```

Fazendo branch no git

Veja que quando voltamos para Branch “main” não aparece o Branch “testando”.

Git CMD - "C:\Program Files\Git\cmd\git.exe" log

```
C:\github\Palomakoba\git_test>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\github\Palomakoba\git_test>git log
commit 53e2097e4af59a08541eae9dcd3fe22bf77fc0d6 (HEAD -> main, origin/main, origin/HEAD)
Author: pablonv <pablo.vargas@unir.br>
Date:   Tue Jul 26 12:00:59 2022 -0400

    Add nome do aluno

commit 2a6ce717d7e57806cbc873b22db3b200afad526d
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Sun Jul 10 09:37:56 2022 -0400

    Add info equipe

commit d16e532236a3007abc6294929fa8d3eb2d5b58e9
```

Fazendo branch no git

Altere novamente o arquivo olaMundo e adicione o commit no branch “main”

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Bem-vindo ao projeto Palomakoba!
Vamos aprender os fundamentos do git?
Claro!
Qual é seu nome aluno?
Ciclano de Tal
```

Git CMD - "C:\Program Files\Git\cmd\git.exe" log

```
C:\github\Palomakoba\git_test>git commit -am "Add mensagem de Bem-vindo"
[main ed96ae3] Add mensagem de Bem-vindo
 1 file changed, 3 insertions(+), 1 deletion(-)

C:\github\Palomakoba\git_test>git log
commit ed96ae366d402a64832804eb8bbaa2b4967b6541 (HEAD -> main)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Wed Jul 27 21:07:59 2022 -0400

    Add mensagem de Bem-vindo

commit 53e2097e4af59a08541eae9dcd3fe22bf77fc0d6 (origin/main, origin/HEAD)
Author: pablonv <pablo.vargas@unir.br>
Date:   Tue Jul 26 12:00:59 2022 -0400

    Add nome do aluno
```

A troca de branches muda os arquivos em seu diretório de trabalho

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Vamos aprender os fundamentos do git?
Claro!
Qual é se
Ciclano d
```

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Bem-vindo ao projeto Palomakoba!
Vamos aprender os fundamentos do git?
Claro!
Qual é seu nome aluno?
Ciclano de Tal
```

Fazendo branch no git

Altere novamente o arquivo olaMundo e adicione o commit no branch “main”

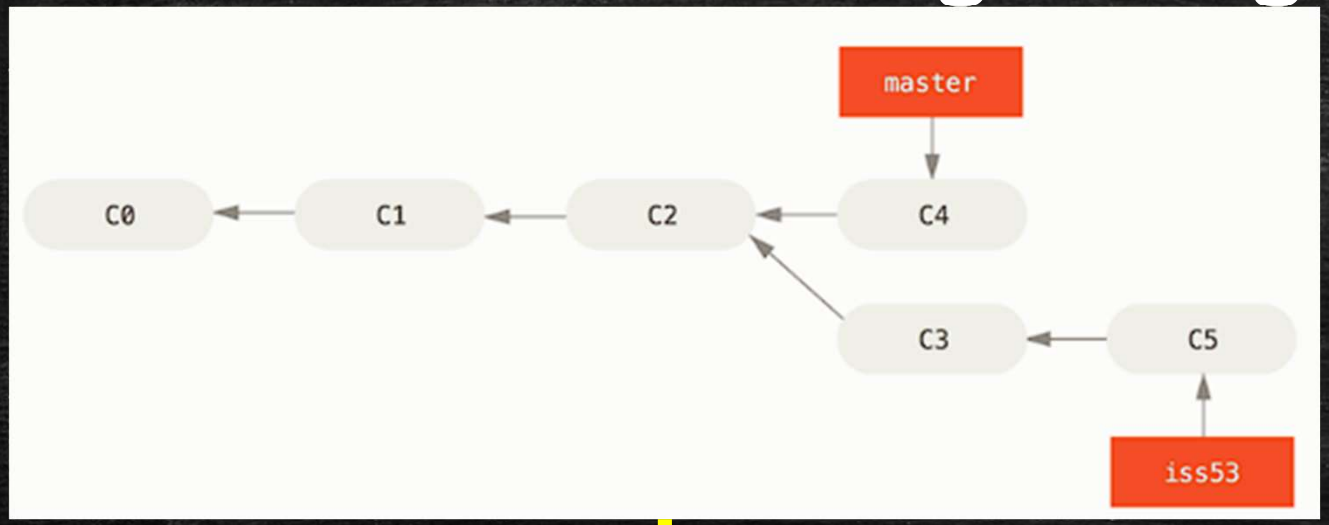
olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Bem-vindo ao projeto Palomakoba!
Vamos aprender os fundamentos do git?
Claro!
Qual é seu nome aluno?
Ciclano de Tal
De qual instituição?
Unir
```

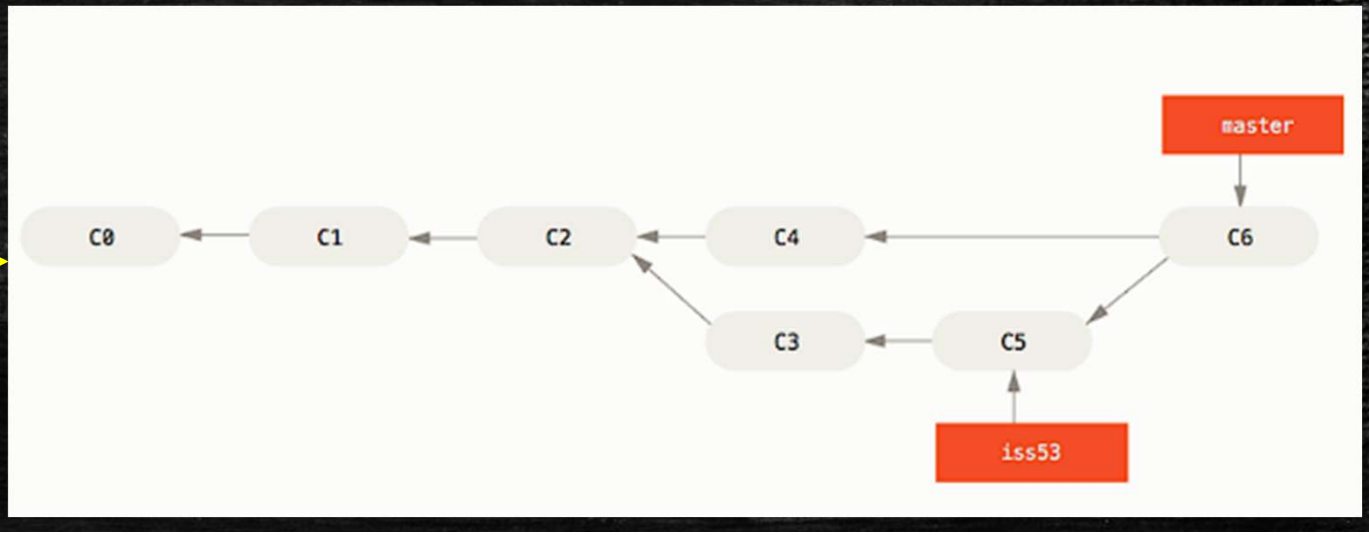
```
C:\github\Palomakoba\git_test>git commit -am "Add nome da instituição"
[main f6479da] Add nome da instituição
 1 file changed, 3 insertions(+), 1 deletion(-)

C:\github\Palomakoba\git_test>git status
On branch main
Your branch is ahead of 'teste/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\github\Palomakoba\git_test>git log --graph
* commit f6479da0e4c140df263c6be35c94fb7f0c2b3995 (HEAD -> main)
| Author: pablonv <pablo.vargas@unir.br>
| Date:   Thu Jul 28 11:42:49 2022 -0400
|
|     Add nome da instituição
```

Fazendo merge no git



Incorpora alterações dos commits nomeados (desde o momento em que seus históricos divergiram da trilha original) na ramificação atual.

Junte dois ou mais históricos de desenvolvimento



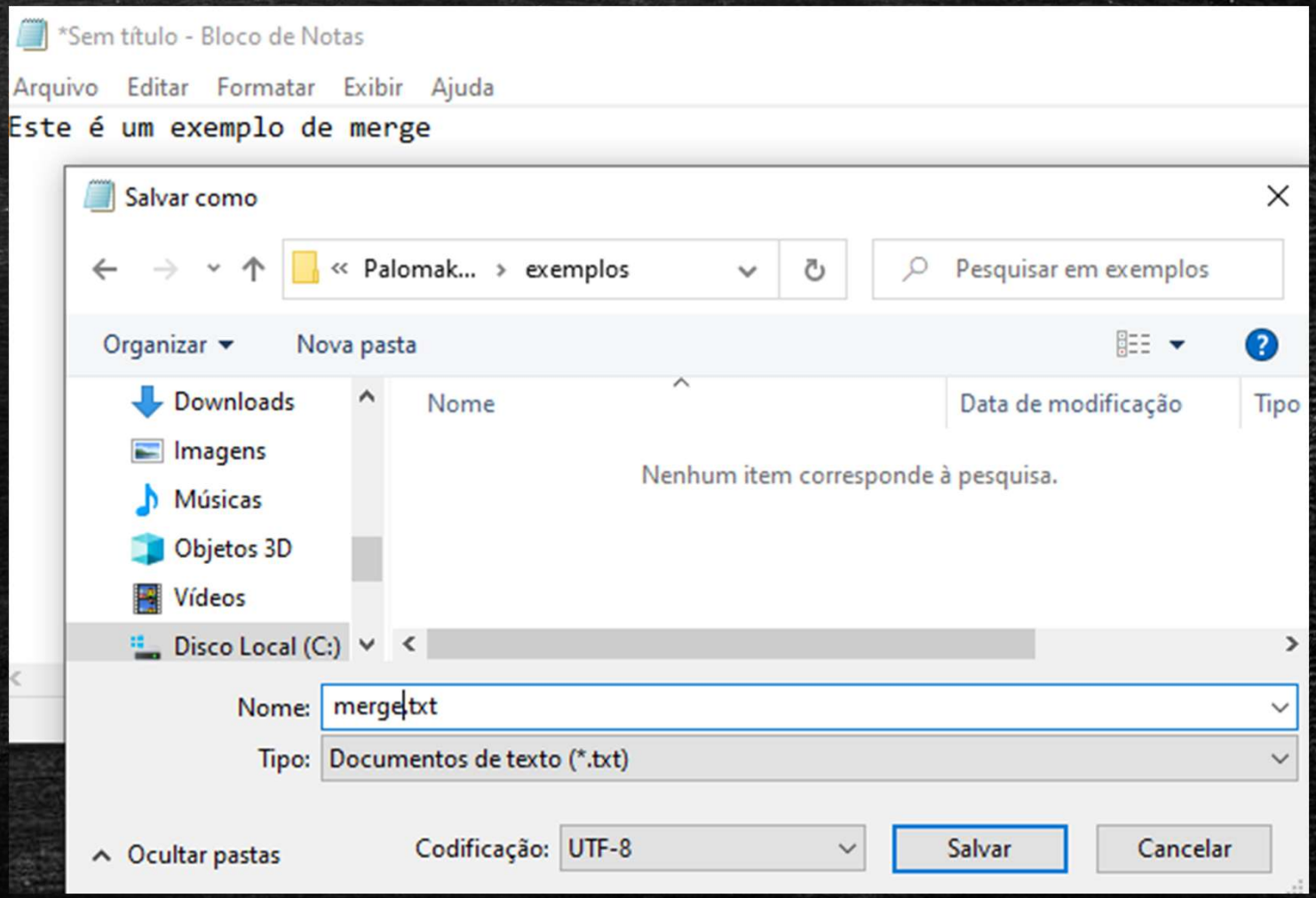
Fazendo merge no git

Crie uma nova pasta em Palomakoba chamada “exemplos” e inicialize um novo repositório git (git init)

```
C:\github\Palomakoba>mkdir exemplos

C:\github\Palomakoba>dir
 O volume na unidade C não tem nome.
 O Número de Série do Volume é 7C0F-DC6A

 Pasta de C:\github\Palomakoba

11/08/2022  09:36    <DIR>          .
11/08/2022  09:36    <DIR>          ..
11/08/2022  09:36    <DIR>          exemplos
28/07/2022  11:31    <DIR>          git_test
08/07/2022  15:06    <DIR>          Meu_libgit
               0 arquivo(s)              0 bytes
               5 pasta(s)   231.077.400.576 bytes disponíveis

C:\github\Palomakoba>cd exemplos

C:\github\Palomakoba\exemplos>git init
Initialized empty Git repository in C:/github/Palomakoba/exemplos/.git/

C:\github\Palomakoba\exemplos>
```

Fazendo merge no git

Crie um arquivo o bloco de notas conforme figura ao lado



Fazendo merge no git

Adicione o arquivo e depois faça um commit

```
C:\github\Palomakoba\exemplos>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        merge.txt

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

C:\github\Palomakoba\exemplos>git add merge.txt

C:\github\Palomakoba\exemplos>git commit -m "Add exemplo merge"
[master (root-commit) 555bac6] Add exemplo merge
 1 file changed, 1 insertion(+)
 create mode 100644 merge.txt

C:\github\Palomakoba\exemplos>
```

Fazendo merge no git

Crie uma branch fulano e troque para essa branch

```
C:\github\Palomakoba\exemplos>git branch fulano
```

```
C:\github\Palomakoba\exemplos>git checkout fulano
Switched to branch 'fulano'
```

Fazendo merge no git

Crie outro arquivo do bloco de notas na pasta exemplos conforme figura ao lado

mergeFulano - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

Este é outro exemplo de merge feito pelo fulano.

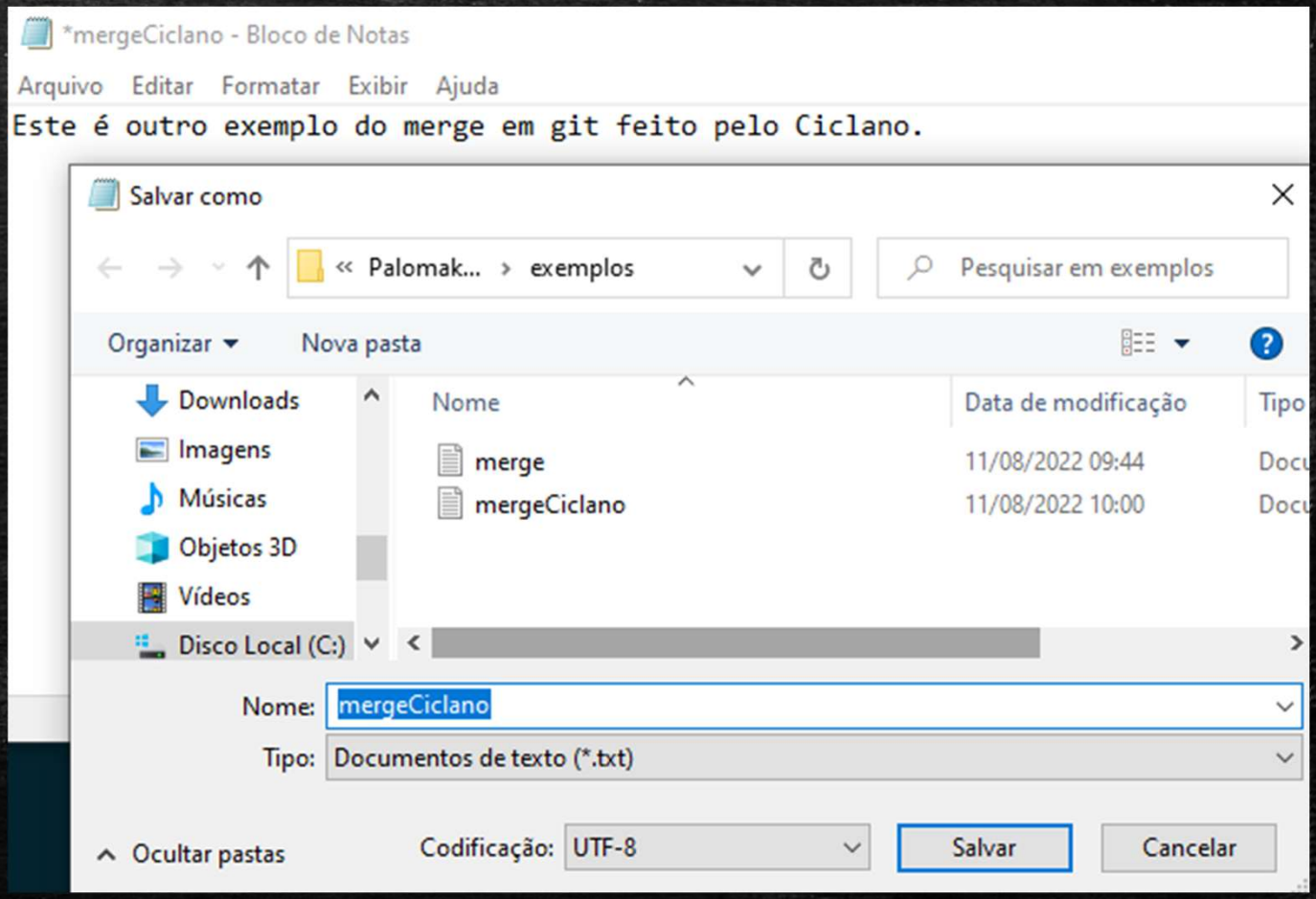
Adicione o arquivo ao repositório e faça um commit nele

```
C:\github\Palomakoba\exemplos>git add mergeFulano.txt

C:\github\Palomakoba\exemplos>git commit -m "Add exemplo merge do fulano"
[fulano 1a4c12a] Add exemplo merge do fulano
 1 file changed, 1 insertion(+)
 create mode 100644 mergeFulano.txt

C:\github\Palomakoba\exemplos>git log
commit 1a4c12ab71559b8909041e255fb86121969a259a (HEAD -> fulano)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 09:52:40 2022 -0400

    Add exemplo merge do fulano

commit 555bac6dd504cc94f85e4ba1c385ae5a9113b2b2 (master)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 09:45:37 2022 -0400

    Add exemplo merge
```

Fazendo merge no git

1º Troque para branch master

2º Crie outro arquivo do bloco de notas na pasta exemplos conforme figura ao lado



Fazendo merge no git

Adicione o arquivo ao repositório e faça um commit nele

```
C:\github\Palomakoba\exemplos>git add mergeCiclano.txt

C:\github\Palomakoba\exemplos>git commit -m "Add exemplo merge do Ciclano"
[master c7bf909] Add exemplo merge do Ciclano
 1 file changed, 1 insertion(+)
 create mode 100644 mergeCiclano.txt

C:\github\Palomakoba\exemplos>git log
commit c7bf909f1fe3a56f2cbf7f80a1b9409a42edc6d8 (HEAD -> master)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 10:02:12 2022 -0400

    Add exemplo merge do Ciclano

commit 555bac6dd504cc94f85e4ba1c385ae5a9113b2b2
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 09:45:37 2022 -0400

    Add exemplo merge
```

Fazendo merge no git

Faça o merge com o branch fulano

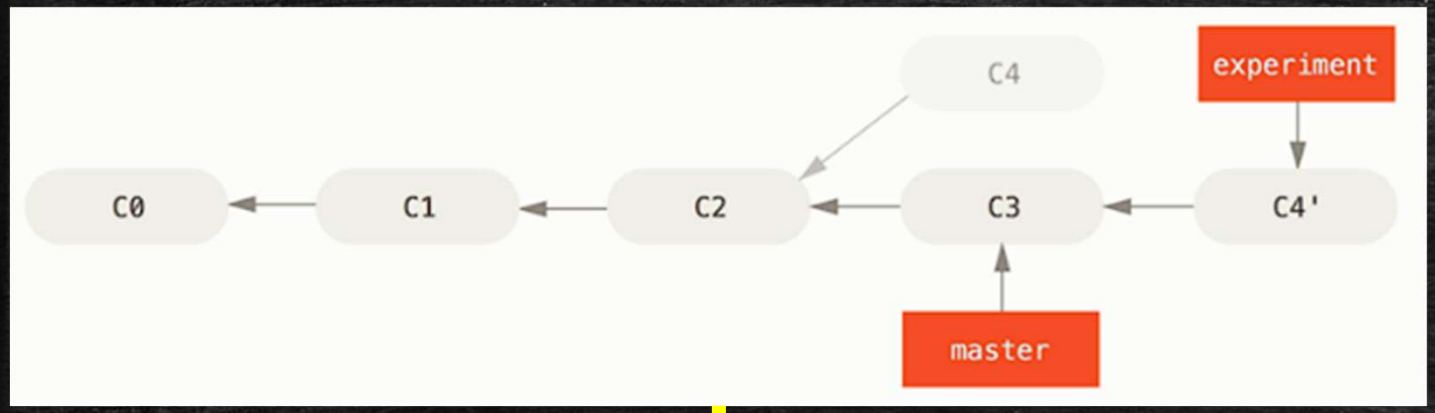
```
C:\github\Palomakoba\exemplos>git branch
  fulano
* master

C:\github\Palomakoba\exemplos>git merge fulano
Merge made by the 'ort' strategy.
 mergeFulano.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 mergeFulano.txt

C:\github\Palomakoba\exemplos>_
```

Fazendo merge no git

```
C:\github\Palomakoba\exemplos>git log --graph
*   commit 9a9badea574ed216fc6af81f75f5729a100f9af5 (HEAD -> master)
|\  Merge: c7bf909 1a4c12a
| | Author: Pablo Vargas <pablo.vargas@unir.br>
| | Date:   Thu Aug 11 10:05:02 2022 -0400
| |
| |     Merge branch 'fulano'
| |
| * commit 1a4c12ab71559b8909041e255fb86121969a259a (fulano)
| | Author: Pablo Vargas <pablo.vargas@unir.br>
| | Date:   Thu Aug 11 09:52:40 2022 -0400
| |
| |     Add exemplo merge do fulano
| |
* | commit c7bf909f1fe3a56f2cbf7f80a1b9409a42edc6d8
|/  Author: Pablo Vargas <pablo.vargas@unir.br>
|   Date:   Thu Aug 11 10:02:12 2022 -0400
|
|       Add exemplo merge do Ciclano
|
* commit 555bac6dd504cc94f85e4ba1c385ae5a9113b2b2
  Author: Pablo Vargas <pablo.vargas@unir.br>
  Date:   Thu Aug 11 09:45:37 2022 -0400

      Add exemplo merge
```

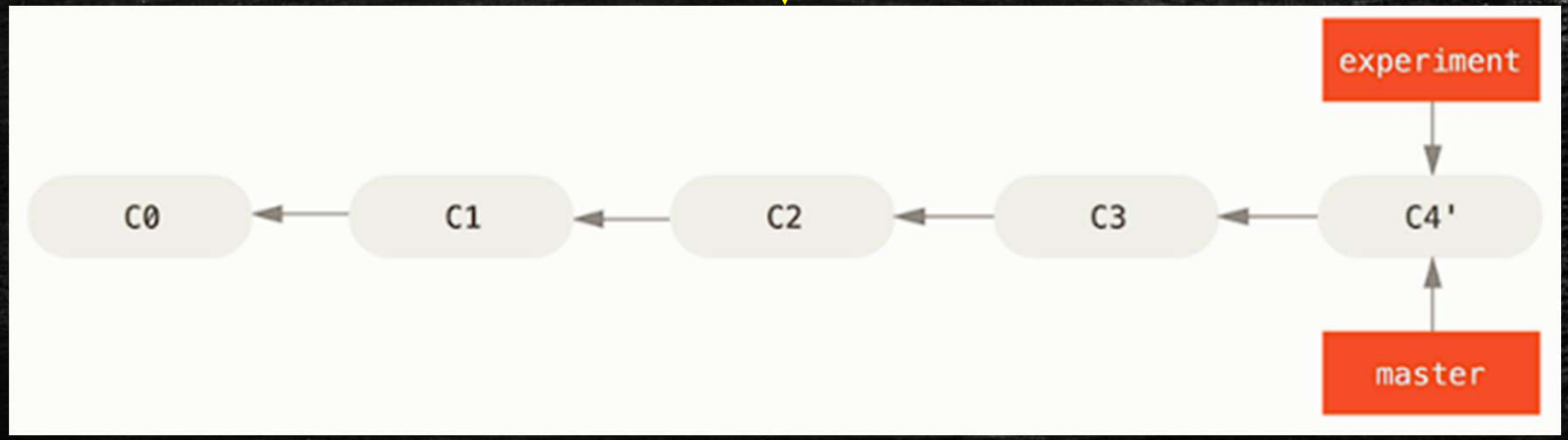
Digite git log --graph para visualizar a ramificação e a mesclagem

Fazendo rebase no git



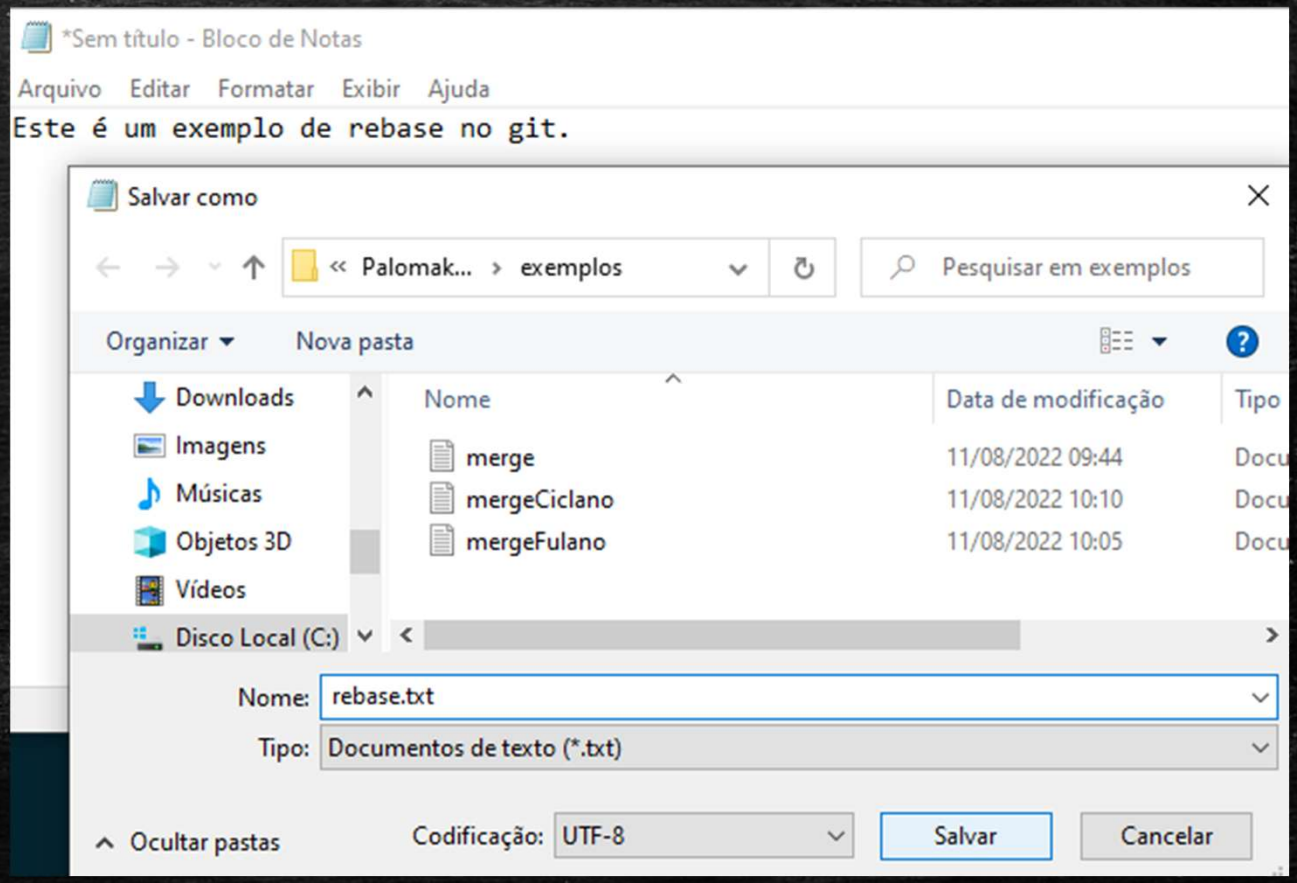
Você pode pegar todas as alterações que foram confirmadas em uma ramificação e reproduzi-las em uma ramificação diferente

Reaplica commits em cima de outra branch



Fazendo rebase no git

Crie outro arquivo do bloco de notas na pasta exemplos conforme figura ao lado



Fazendo rebase no git

Adicione o arquivo ao repositório e faça um commit nele dentro do master

```
C:\github\Palomakoba\exemplos>git add rebase.txt

C:\github\Palomakoba\exemplos>git commit -m "Add exemplo rebase"
[master aad9df8] Add exemplo rebase
 1 file changed, 1 insertion(+)
 create mode 100644 rebase.txt

C:\github\Palomakoba\exemplos>git log
commit aad9df84af892a64b297540f3c553102a55141d1 (HEAD -> master)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 10:13:53 2022 -0400

    Add exemplo rebase

commit 9a9badea574ed216fc6af81f75f5729a100f9af5
Merge: c7bf909 1a4c12a
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 10:05:02 2022 -0400

    Merge branch 'fulano'

commit c7bf909f1fe3a56f2cbf7f80a1b9409a42edc6d8
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 10:02:12 2022 -0400
```

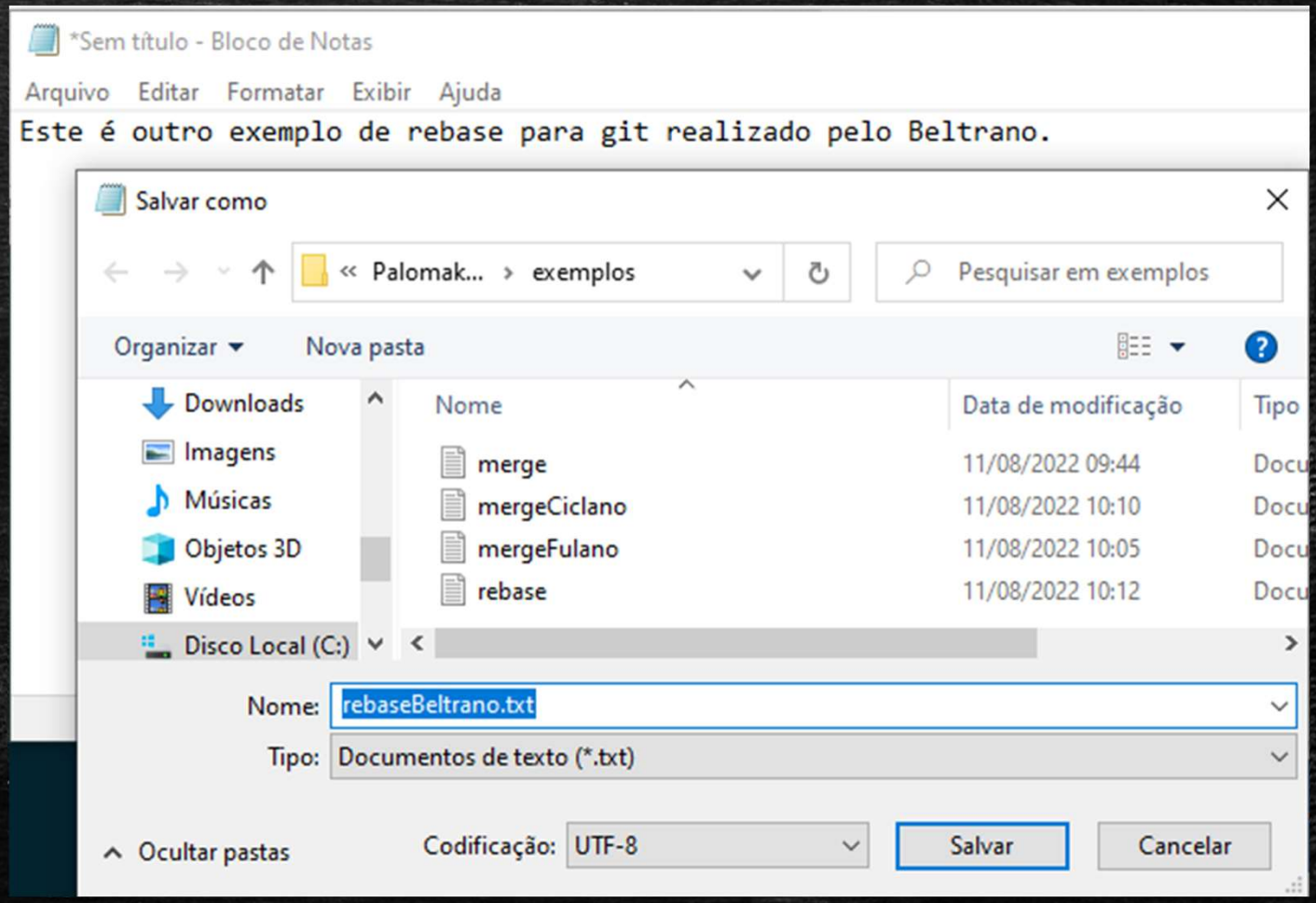
Fazendo rebase no git

Crie um novo branch chamado de beltrano e mude para essa branch

```
C:\github\Palomakoba\exemplos>git checkout -b beltrano
Switched to a new branch 'beltrano'

C:\github\Palomakoba\exemplos>git branch
* beltrano
  fulano
  master

C:\github\Palomakoba\exemplos>
```

Fazendo rebase no git

Crie outro arquivo do bloco de notas na pasta exemplos conforme figura ao lado



Fazendo rebase no git

Adicione o arquivo ao repositório e faça um commit nele

```
C:\github\Palomakoba\exemplos>git add rebaseBeltrano.txt

C:\github\Palomakoba\exemplos>git commit -m "Add exemplo rebase beltrano"
[beltrano f16dcfe] Add exemplo rebase beltrano
 1 file changed, 1 insertion(+)
 create mode 100644 rebaseBeltrano.txt

C:\github\Palomakoba\exemplos>git log
commit f16dcfec812f0a01962e687c701dc5169a0bb04f (HEAD -> beltrano)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 10:20:41 2022 -0400

    Add exemplo rebase beltrano

commit aad9df84af892a64b297540f3c553102a55141d1 (master)
Author: Pablo Vargas <pablo.vargas@unir.br>
Date:   Thu Aug 11 10:13:53 2022 -0400

    Add exemplo rebase

commit 9a9badea574ed216fc6af81f75f5729a100f9af5
Merge: c7bf909 1a4c12a
```

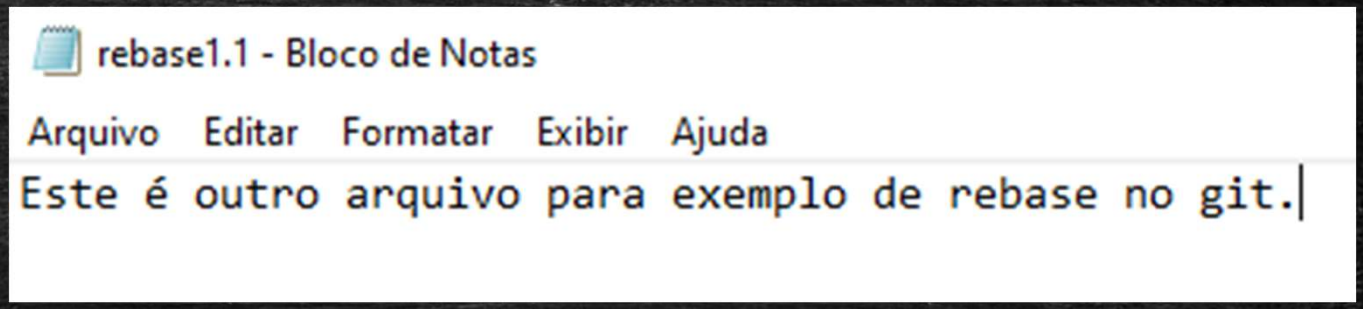
Fazendo rebase no git

Digite git log --graph

```
C:\github\Palomakoba\exemplos>git log --graph
* commit f16dcfec812f0a01962e687c701dc5169a0bb04f (HEAD -> beltrano)
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Thu Aug 11 10:20:41 2022 -0400
|
|     Add exemplo rebase beltrano
|
* commit aad9df84af892a64b297540f3c553102a55141d1 (master)
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Thu Aug 11 10:13:53 2022 -0400
|
|     Add exemplo rebase
|
*   commit 9a9badea574ed216fc6af81f75f5729a100f9af5
|\  Merge: c7bf909 1a4c12a
| | Author: Pablo Vargas <pablo.vargas@unir.br>
| | Date:   Thu Aug 11 10:05:02 2022 -0400
| |
| |     Merge branch 'fulano'
| |
| * commit 1a4c12ab71559b8909041e255fb86121969a259a (fulano)
| | Author: Pablo Vargas <pablo.vargas@unir.br>
| | Date:   Thu Aug 11 09:52:40 2022 -0400
| |
| |     Add exemplo merge do fulano
| |
* | commit c7bf909f1fe3a56f2cbf7f80a1b9409a42edc6d8
```

Fazendo rebase no git

1º Mude para branch master. Crie outro arquivo do bloco de notas na pasta exemplos conforme figura ao lado



2º Adicione o arquivo ao repositório e faça um commit nele

```
C:\github\Palomakoba\exemplos>git add rebase1.1.txt

C:\github\Palomakoba\exemplos>git commit -m "Add exemplo rebase1.1"
[master 70c1b71] Add exemplo rebase1.1
 1 file changed, 1 insertion(+)
 create mode 100644 rebase1.1.txt

C:\github\Palomakoba\exemplos>git log --graph
* commit 70c1b711a8efd97d87afb459bc969ebb0a11ccba (HEAD -> master)
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Thu Aug 11 10:28:38 2022 -0400
|
|     Add exemplo rebase1.1
|
* commit aad9df84af892a64b297540f3c553102a55141d1
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Thu Aug 11 10:13:53 2022 -0400
|
|     Add exemplo rebase
|
*   commit 9a9badea574ed216fc6af81f75f5729a100f9af5
|\  Merge: c7bf909 1a4c12a
| | Author: Pablo Vargas <pablo.vargas@unir.br>
| | Date:   Thu Aug 11 10:05:02 2022 -0400
```

Fazendo rebase no git

Faça o "git rebase beltrano" para integrar o branch de maneira linear

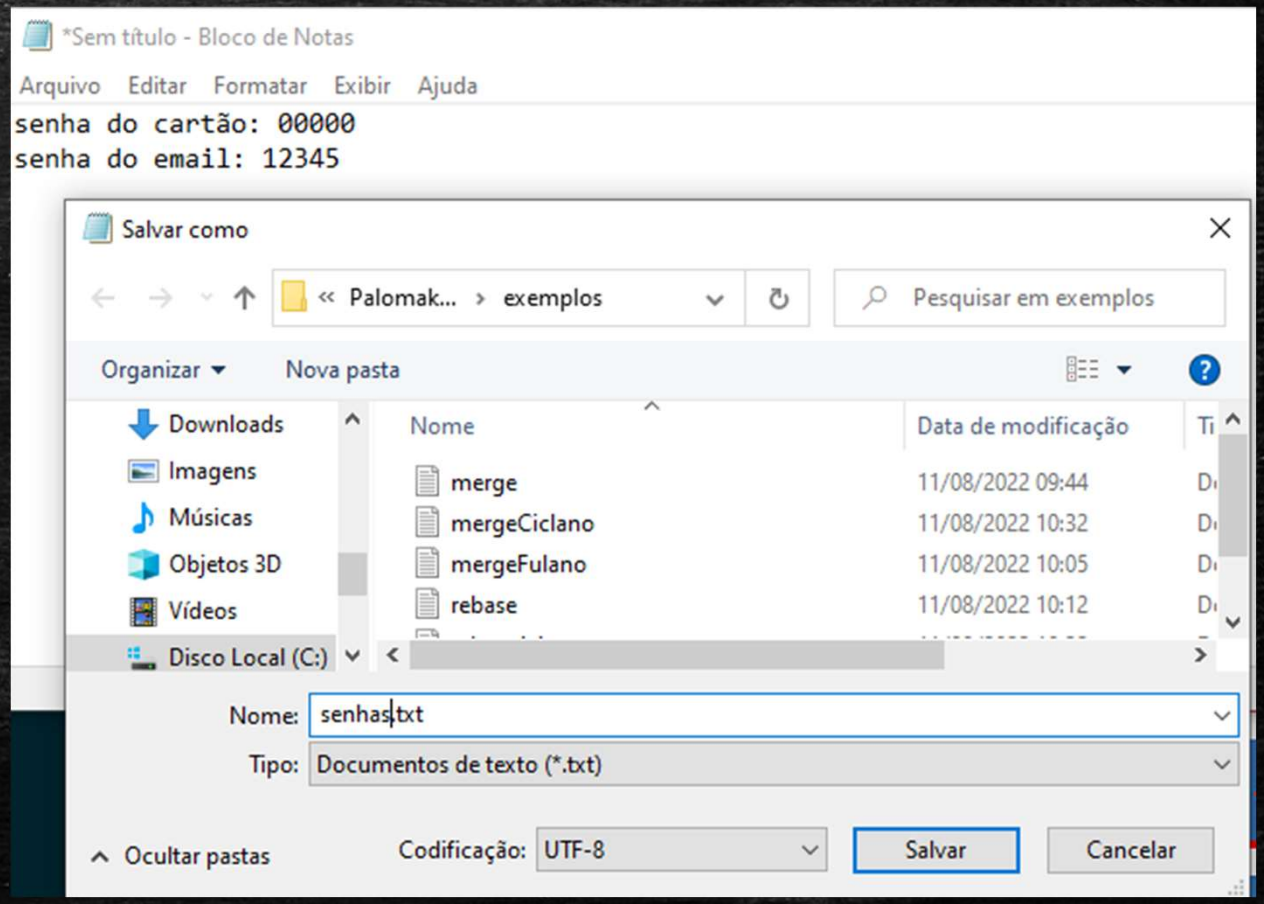
```
C:\github\Palomakoba\exemplos>git rebase beltrano
Successfully rebased and updated refs/heads/master.

C:\github\Palomakoba\exemplos>git log --graph
* commit e82f2a5da98b687ecef8d27aae86b60058696046 (HEAD -> master)
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Thu Aug 11 10:28:38 2022 -0400
|
|     Add exemplo rebase1.1
|
* commit f16dcfec812f0a01962e687c701dc5169a0bb04f (beltrano)
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Thu Aug 11 10:20:41 2022 -0400
|
|     Add exemplo rebase beltrano
|
* commit aad9df84af892a64b297540f3c553102a55141d1
| Author: Pablo Vargas <pablo.vargas@unir.br>
| Date:   Thu Aug 11 10:13:53 2022 -0400
|
|     Add exemplo rebase
|
*   commit 9a9badea574ed216fc6af81f75f5729a100f9af5
|\  Merge: c7bf909 1a4c12a
| | Author: Pablo Vargas <pablo.vargas@unir.br>
| | Date:   Thu Aug 11 10:05:02 2022 -0400
| |
```

.gitignore

Crie um arquivo do bloco de notas na pasta exemplos conforme figura abaixo

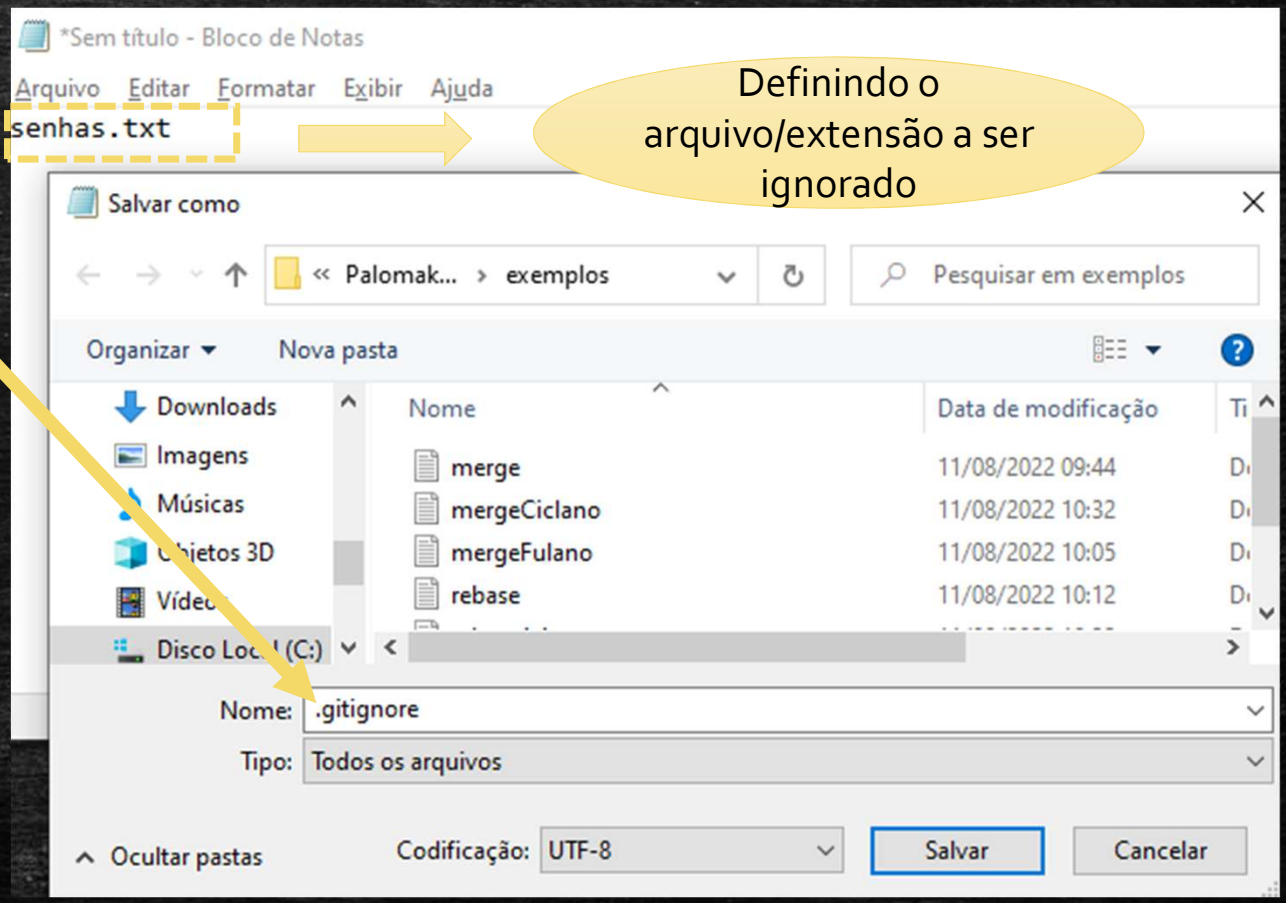
Especifica arquivos não rastreados intencionalmente que o Git deve ignorar.

Interessante para não monitorar arquivos confidenciais e privados como senhas, configurações..



.gitignore

Crie outro arquivo do bloco de notas na pasta exemplos conforme figura ao lado



Obs: pode-se ignorar tanto um arquivo específico como uma extensão.

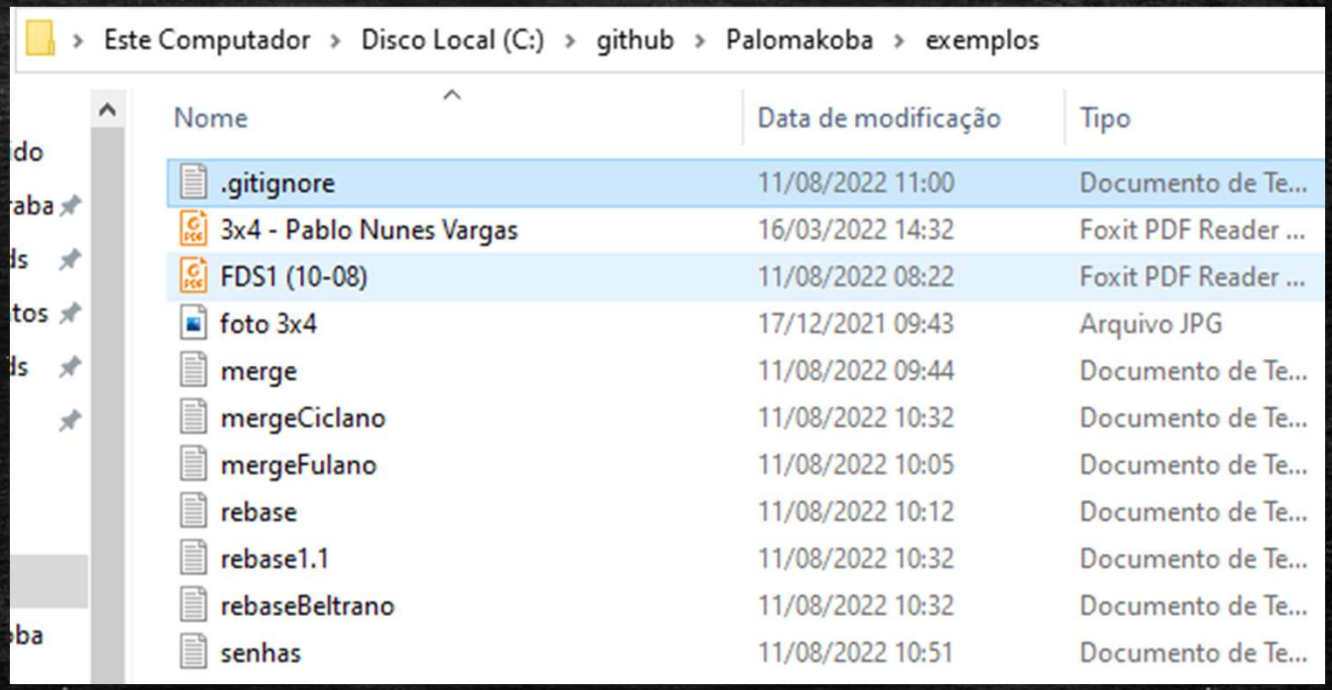
.gitignore

Ao digitar “git status” o arquivo senhas.txt é ignorado pelo git

```
C:\github\Palomakoba\exemplos>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

C:\github\Palomakoba\exemplos>
```

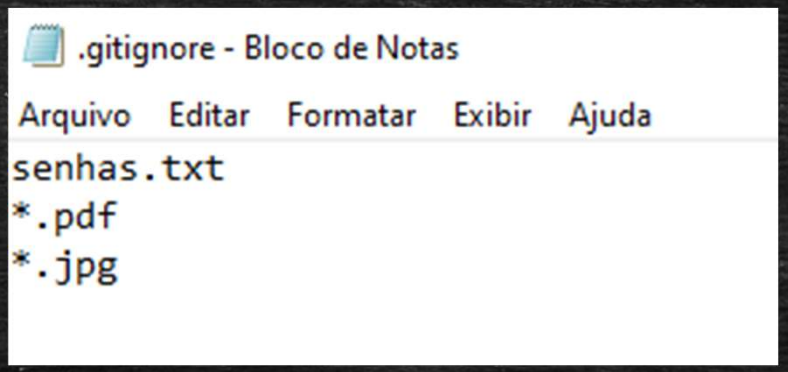
.gitignore

Adicione alguns arquivos com extensão pdf e jpg



.gitignore

Modifique o arquivo .gitignore conforme figura ao lado



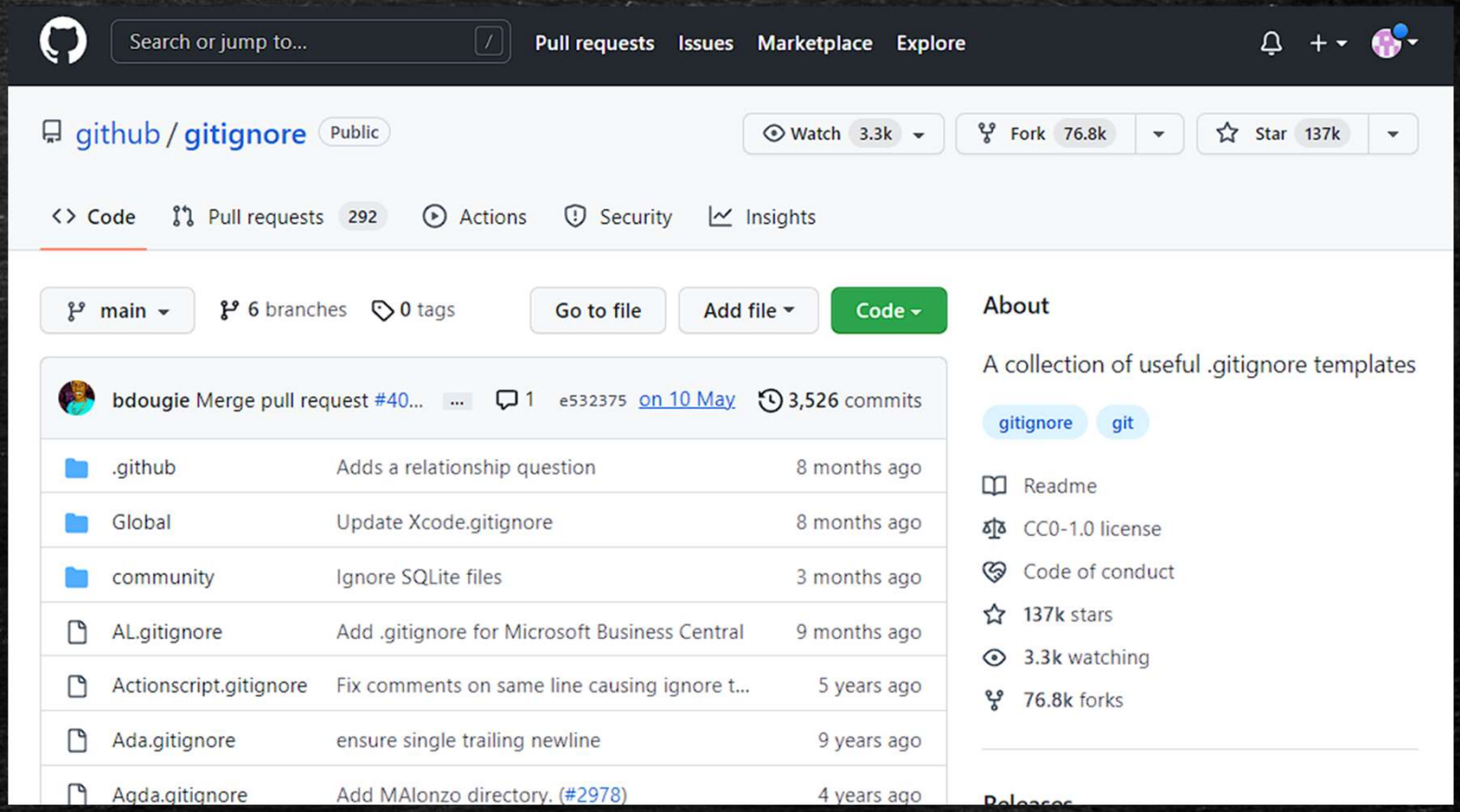
Digite "git status" e veja que o git ignora os arquivos .pdf e .jpg também

```
C:\github\Palomakoba\exemplos>git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

C:\github\Palomakoba\exemplos>
```

.gitignore

https://github.com/github/gitignore

Possui vários templetes para ignorar arquivos de configuração e construção dispensáveis



stash

O comando "git stash" acumula as alterações em um outro diretório de trabalho.

Edite o arquivo olaMundo.txt

olaMundo - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Olá, Mundão!
Eu sou o Pablo Nunes Vargas.
Bem-vindo ao projeto Palomakoba!
Vamos aprender os fundamentos do git?
Claro!
Qual é seu nome aluno?
Ciclano de Tal
De qual instituição?
Unir
Quantos anos?
20
```

stash

Veja que após "git stash" **armazena-se temporariamente** aquela modificação em outro repositório.

Esse tipo de comando é útil quando estamos trabalhando em uma arquivo e então surge uma demanda para outro arquivo. Diante disso, talvez você não queira criar commit daquele arquivo inicial ainda.

```
C:\github\Palomakoba\git_test>git status
On branch Informações_Pessoais
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   olaMundo.txt

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

C:\github\Palomakoba\git_test>git stash
Saved working directory and index state WIP on Informações_Pessoais: f6479da Add nome da instituição

C:\github\Palomakoba\git_test>git status
On branch Informações_Pessoais
nothing to commit, working tree clean

C:\github\Palomakoba\git_test>
```

stash

O "git stash apply" é um comando para arquivo retornar do repositório temporário.

```
C:\github\Palomakoba\git_test>git stash apply
On branch Informações_Pessoais
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   olaMundo.txt

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

O "git stash list" é um comando para listar stash.

```
C:\github\Palomakoba\git_test>git stash list
stash@{0}: WIP on Informações_Pessoais: f6479da Add nome da instituição

C:\github\Palomakoba\git_test>git stash clear

C:\github\Palomakoba\git_test>git stash list
```

O "git stash clear" é um comando para limpar os stash.

alias

São **atalhos** para comandos do git

```
C:\github\Palomakoba\git_test>git config --global alias.s status

C:\github\Palomakoba\git_test>git s
On branch Informações_Pessoais
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   olaMundo.txt

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

C:\github\Palomakoba\git_test>_
```

Exercício: crie um alias para o comando commit

alias

```
C:\github\Palomakoba\git_test>git config --global alias.ct commit

C:\github\Palomakoba\git_test>
```

```
C:\github\Palomakoba\git_test>
[Informações_Pessoais e1b2e36] Add idade aluno
 1 file changed, 3 insertions(+), 1 deletion(-)

C:\github\Palomakoba\git_test>git status
On branch Informações_Pessoais
nothing to commit, working tree clean
```

git ct –am “Add idade aluno”

alias

Você pode **remover** um alias executando “git config --global **--unset** alias.NAME”

tags

Serve para **marcar pontos** específicos **na cronologia** do arquivo como sendo importantes. Normalmente as pessoas usam essa funcionalidade para marcar pontos onde foram feitas *releases* (ex: v1.0).

Parametros: **-a** você adiciona a tag
–m uma mensagem descrevendo a tag

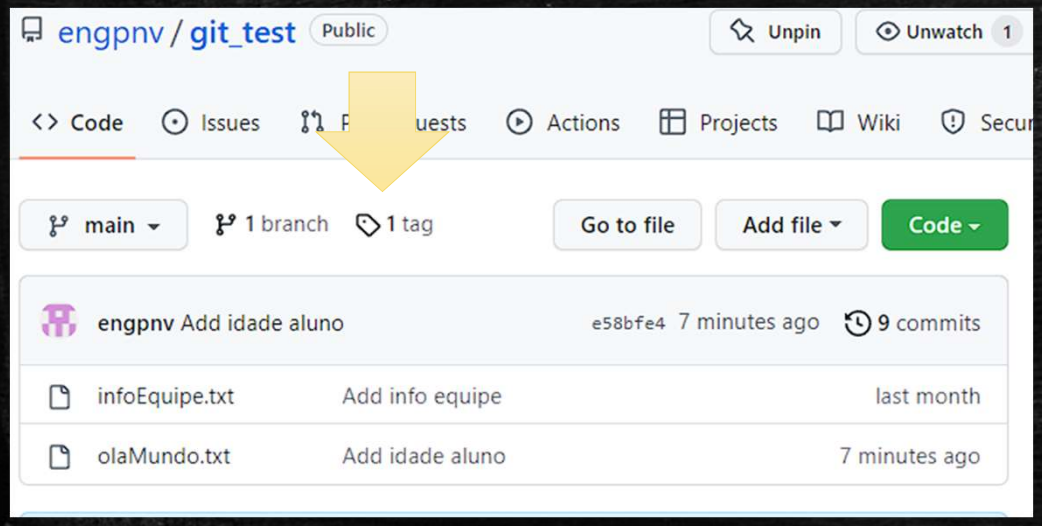
```
C:\github\Palomakoba\git_test>git tag -a 1.0.0 -m "Primeira versão finalizada"

C:\github\Palomakoba\git_test>git tag
1.0.0
```

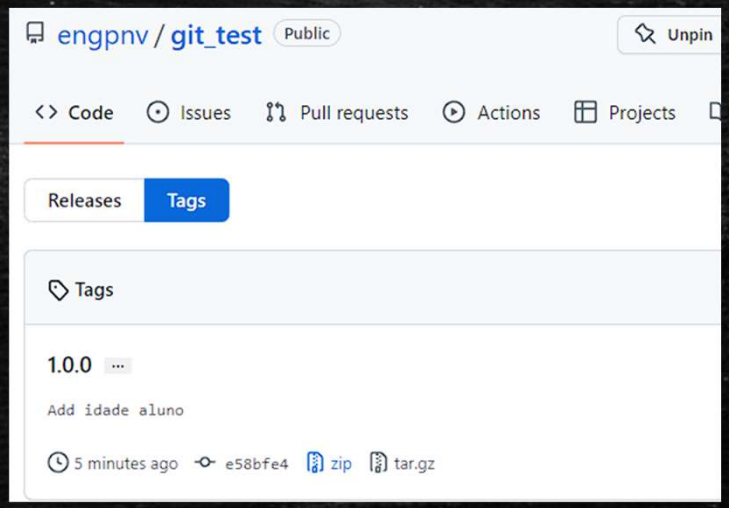
Lista as tags existentes

tags

O “git push origin main --tags” envia para repositório remoto as tags.

```
C:\github\Palomakoba\git_test>git push origin main --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 161 bytes | 161.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/engpnv/git_test.git
 * [new tag]         1.0.0 -> 1.0.0
```





Apagando tags e branches

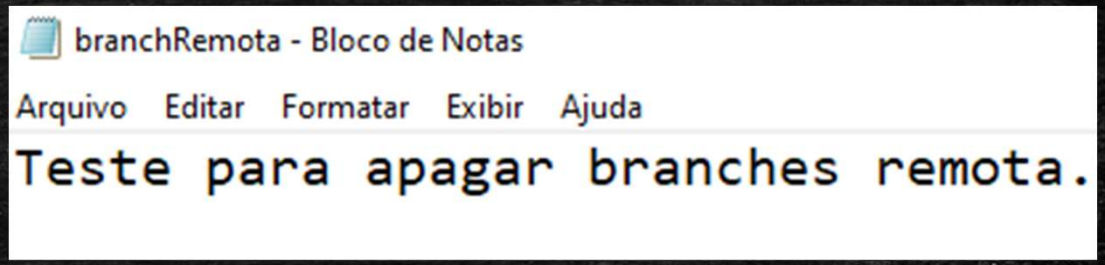
Apagando no **repositório local**:
"git tag –d nomeDaTag"
"git branch –d nomeDaBranch"

```
C:\github\Palomakoba\git_test>git tag
1.0.0

C:\github\Palomakoba\git_test>git tag -d 1.0.0
Deleted tag '1.0.0' (was 6ca2618)

C:\github\Palomakoba\git_test>git tag

C:\github\Palomakoba\git_test>git branch
  Informações_Pessoais
* main

C:\github\Palomakoba\git_test>git branch -d Informações_Pessoais
Deleted branch Informações_Pessoais (was f6479da).

C:\github\Palomakoba\git_test>git branch
* main
```

Apagando tags e branches

Apagando no **repositório remoto**:
"git push origin :nomeDaTag"
"git push origin :nomeDaBranch"

Antes, crie um arquivo e adicione ao repositório local



Apagando tags e branches

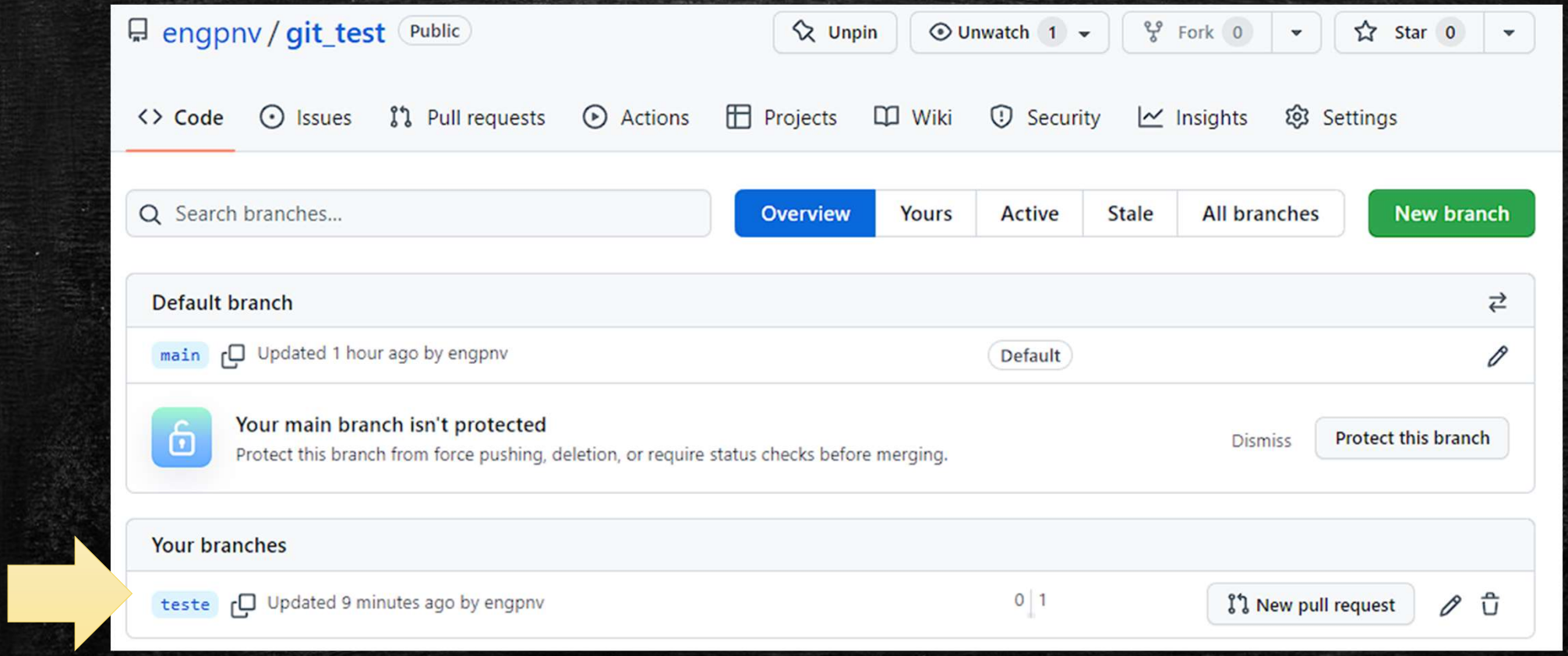
1 º Adicione o arquivo no git e faça um commit nele

```
C:\github\Palomakoba\git_test>git add branchRemota.txt

C:\github\Palomakoba\git_test>git ct -am "Add teste apagar branch remota"
[teste f46dc1a] Add teste apagar branch remota
 1 file changed, 1 insertion(+)
 create mode 100644 branchRemota.txt
```

2º Adicione a branch teste ao remoto git_test

```
C:\github\Palomakoba\git_test>git push origin teste
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 363 bytes | 363.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'teste' on GitHub by visiting:
remote:      https://github.com/engpnv/git_test/pull/new/teste
remote:
To https://github.com/engpnv/git_test.git
 * [new branch]      teste -> teste
```

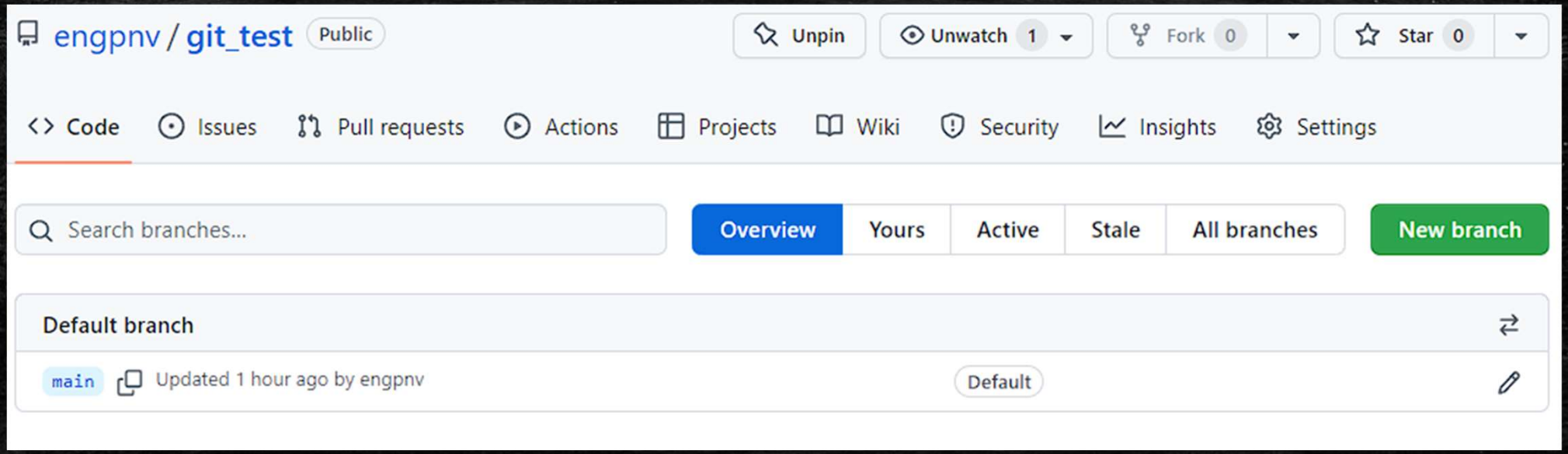
Apagando tags e branches



Apagando tags e branches

Delete o branch teste do repositório remoto

```
C:\github\Palomakoba\git_test>git push origin :teste
To https://github.com/engpnv/git_test.git
 - [deleted]         teste
```



Avaliação

Faça um merge e um alias do reset usando uma nova pasta e um outro arquivo

Envie imagens do git log --graph para o merge e depois uma da operação usando o alias

Referências

- https://www.youtube.com/watch?v=ZDo_f3ZibFA
- https://docs.github.com/en/get-started/using-git/about-git
- https://asana.com/pt/resources
- PRESSMAN, R. S. *Engenharia de Software.* 7ª Edição. McGraw Hill, 2010.
- JUNIOR, H. E. *Engenharia de Software na Prática.* 2ª Edição. São Paulo: Novatec, 2010.
- SOMMERVILLE, I. *Engenharia de Software*. 9ª Edição. São Paulo: Pearson Prentice Hall, 2013.

Referências

- http://pt.wikipedia.org/wiki/Desenvolvimento_%C3%A1gil_de_software
- https://git-scm.com/book/pt-br/v2
- https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf
- https://www.meistertask.com/pt
- HIRAMA, K. *Engenharia de Software: qualidade e produtividade com tecnologia*. 1ª Edição. Rio de Janeiro: Elsevier, 2011.