# GUIA PRÁTICO DE GIT E GITHUB

Daniele Santiago

# DANIELE SANTIAGO

Técnica em Informática pelo IFBA e Graduanda em **Ciências da Computação** pela **UFSCar**

Estagiária em **Ciências de Dados na Nubank**

Ex-mentorada e aluna do **EBA**

Monitora de **Probabilidade e Estatística** na UFSCar

Auxilio na produção de conteúdo dentro do **PED**

Já  atuei como **Desenvolvedora Web** na Liga de Mercado Financeiro da UFSCar e no IFBA Eunápolis

P.E.D

1
INTRODUÇÃO AO GIT E GITHUB

2
USANDO GITHUB SEM TERMINAL

3
COMANDOS ESSENCIAIS DO GIT

4
USANDO GIT NO TERMINAL

5
GITHUB NO MUNDO CORPORATIVO

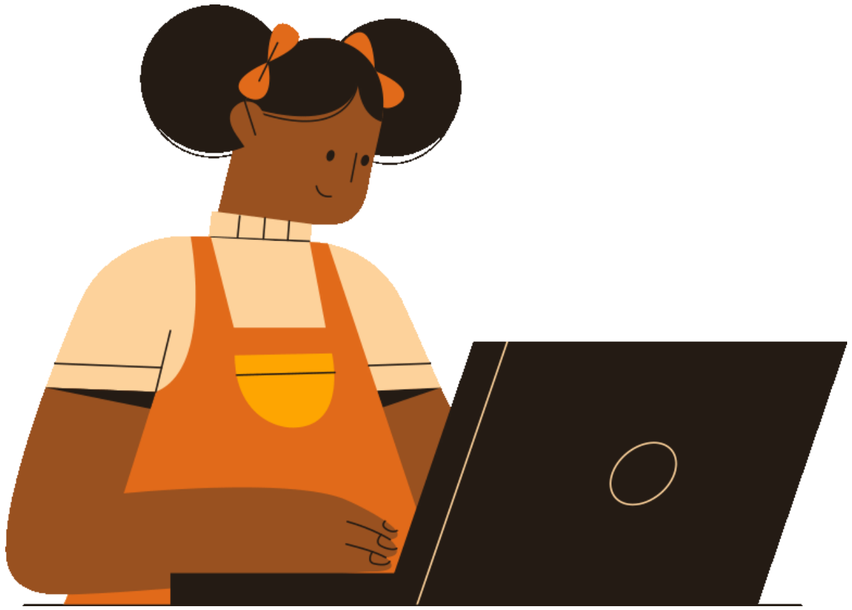
# INTRODUÇÃO AO GIT E GITHUB

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# O QUE É GIT?

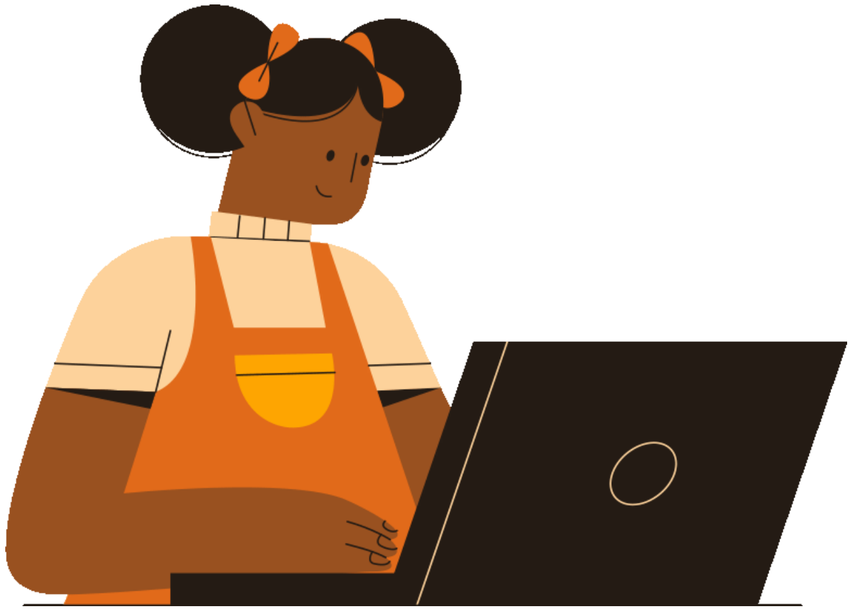
- **Git é um sistema de controle de versão, utilizado para desenvolvimento de software.**
- Ele é conhecido como um Repositório Local.
- Registra cada operação com um “commit”.
- Os projetos Git podem ser hospedados em plataformas como o GitHub.

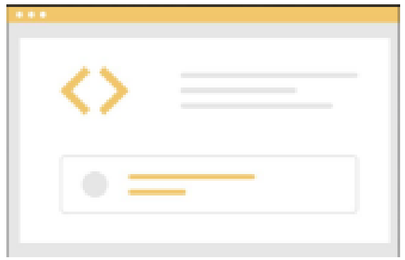
# SEM O GIT



Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

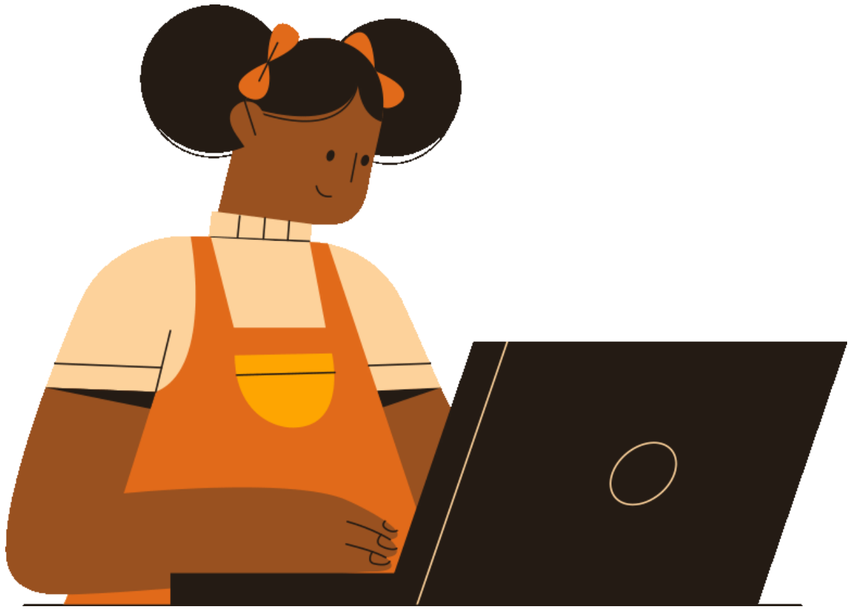
# SEM O GIT

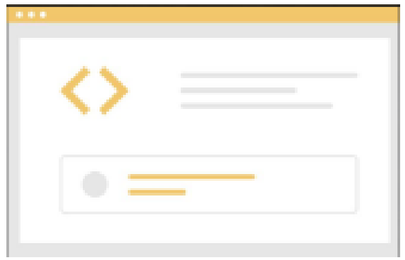




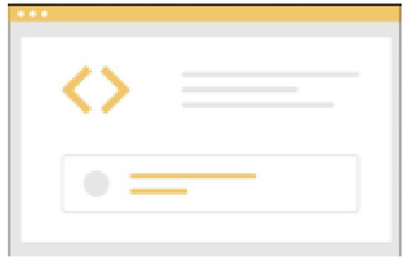
Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# SEM O GIT

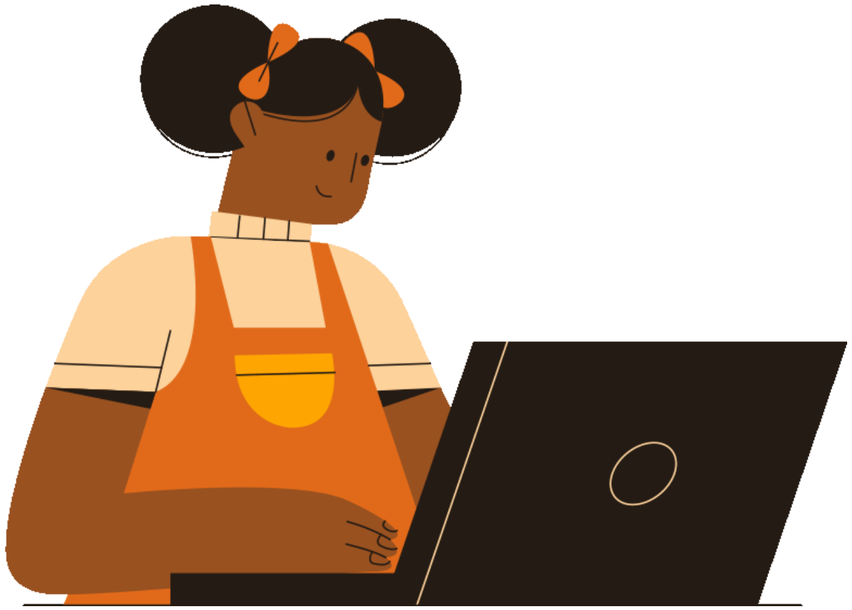


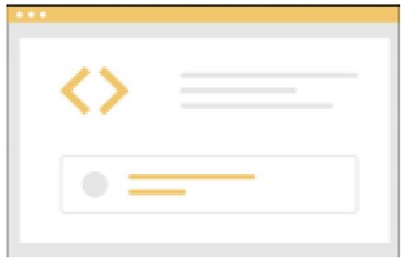


Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

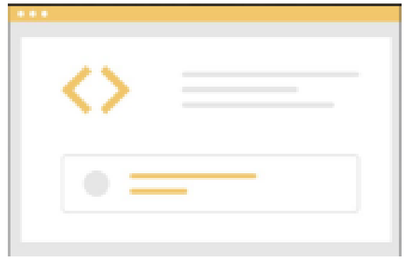


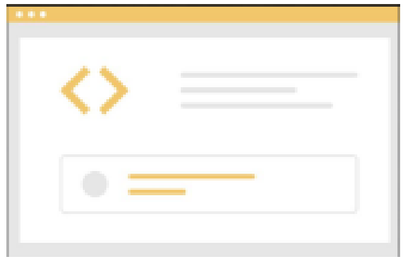
# SEM O GIT



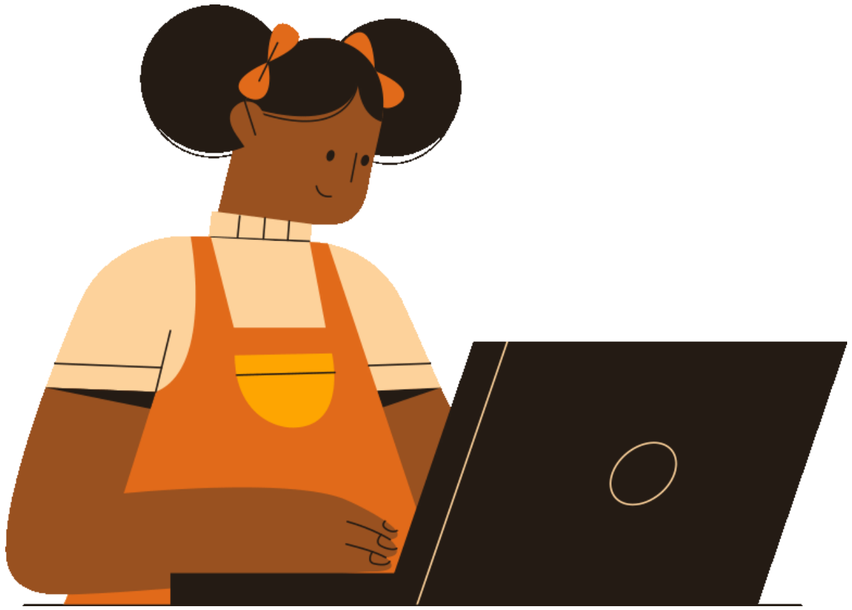


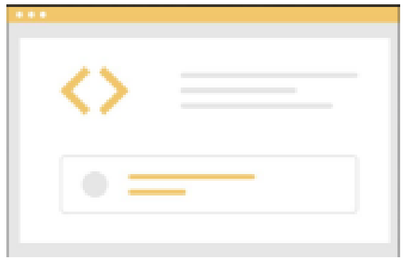
Julia Sara de Aragao Sanda
juliansanda2020@gmail.com



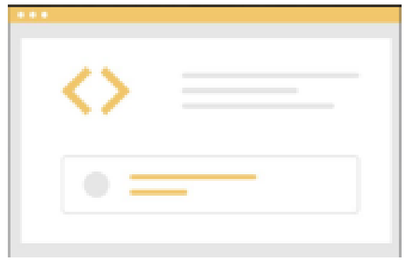


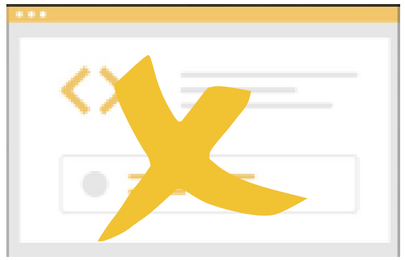
# SEM O GIT



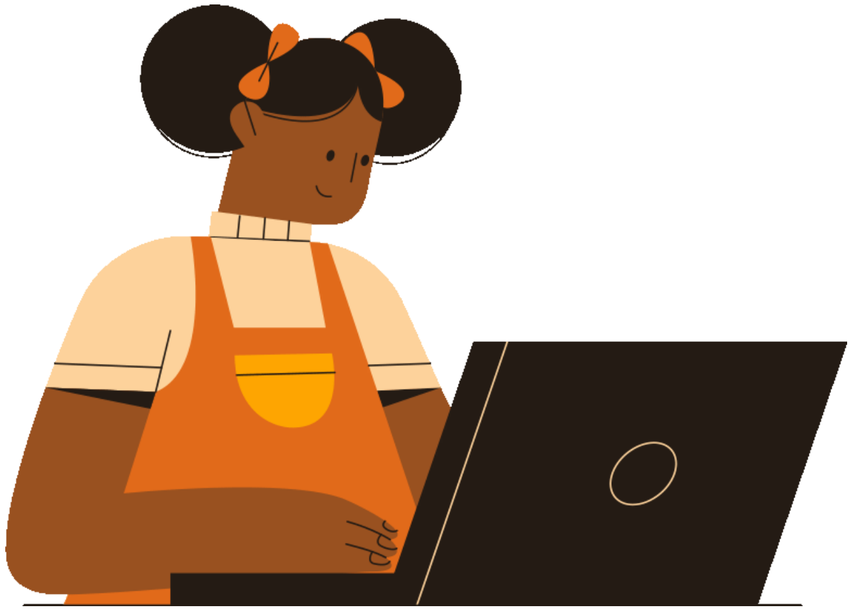


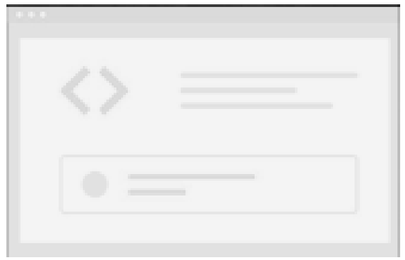
Julia Sara de Aragao Sanda
juliansanda2020@gmail.com



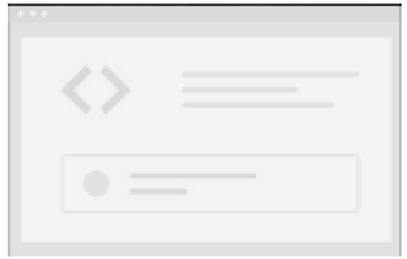


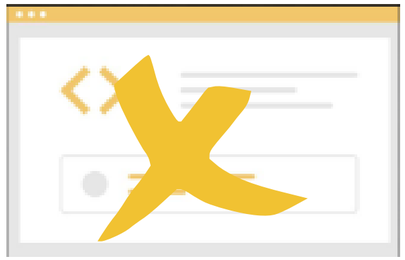
# SEM O GIT



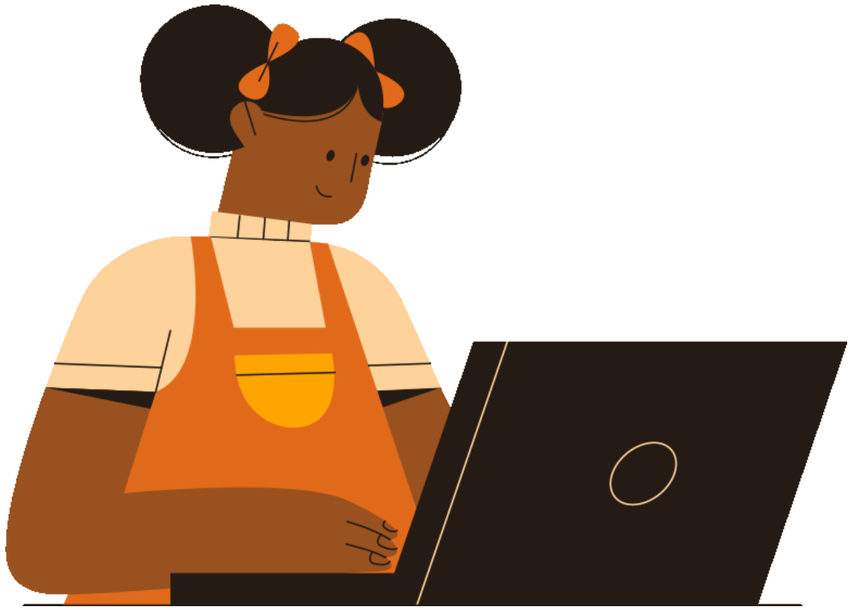


Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

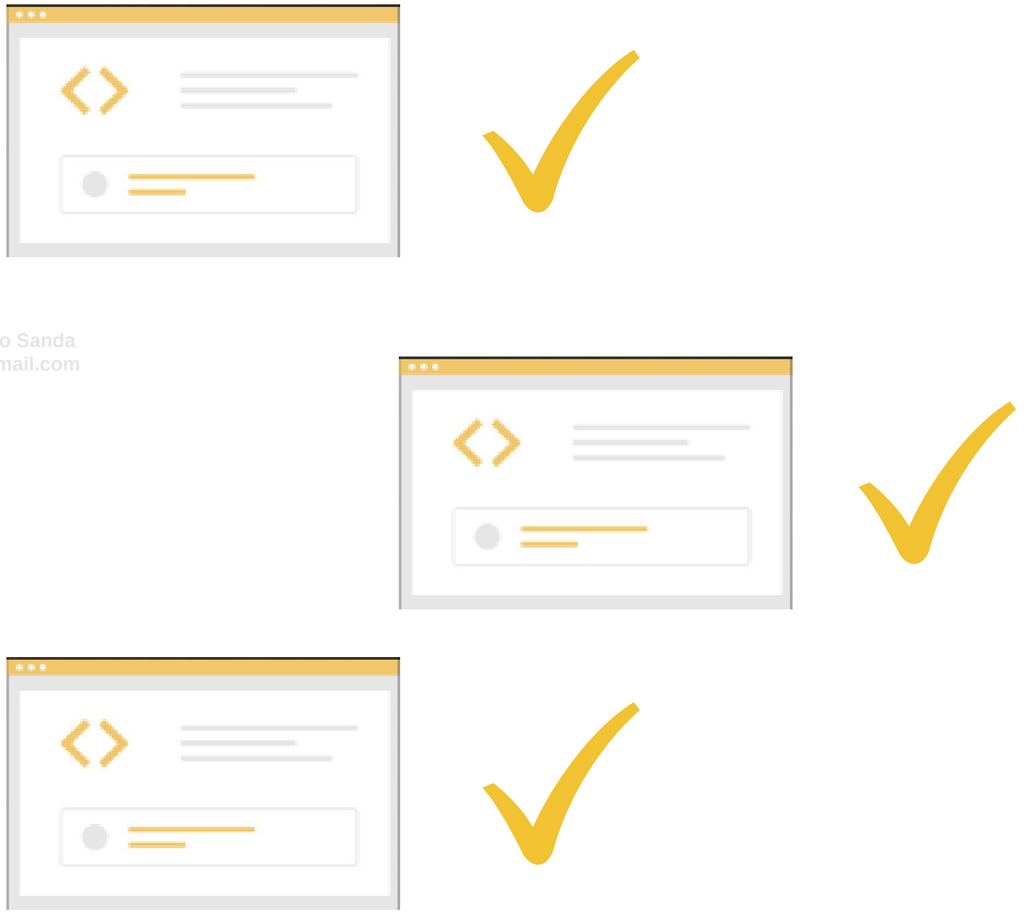




# COM O GIT



Julia Sara de Aragao Sanda
juliansanda2020@gmail.com



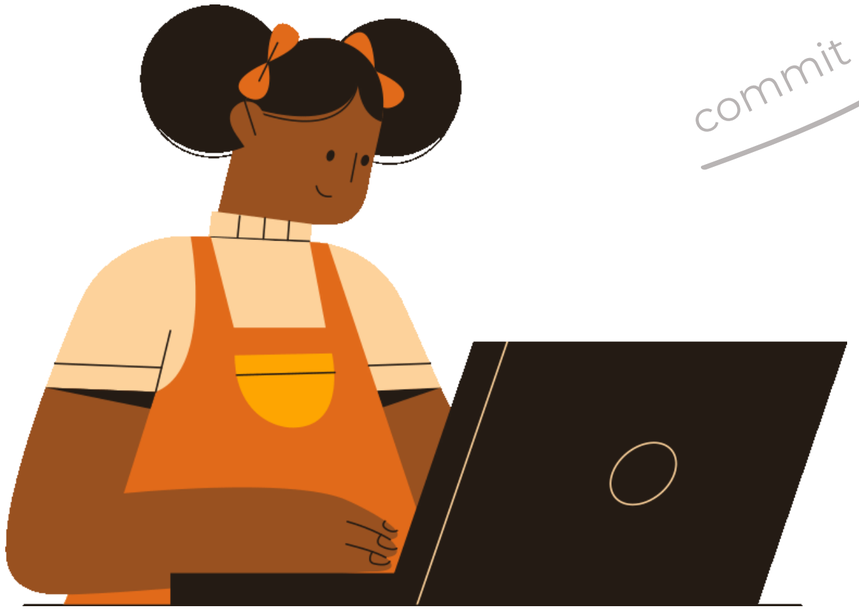
# O QUE É GIT?

- Git é um sistema de controle de versão, utilizado para desenvolvimento de software.
- **Ele é conhecido como um Repositório Local.**
- Registra cada operação com um “commit”.
- Os projetos Git podem ser hospedados em plataformas como o GitHub.

# O QUE É GIT?

- Git é um sistema de controle de versão, utilizado para desenvolvimento de software.
- Ele é conhecido como um Repositório Local.
- **Registra cada operação com um “commit”.**
- Os projetos Git podem ser hospedados em plataformas como o GitHub.

# COM O GIT



commit

commit

commit

# O QUE É GIT?

- Git é um sistema de controle de versão, utilizado para desenvolvimento de software.
- Ele é conhecido como um Repositório Local.
- Registra cada operação com um “commit”.
- **Os projetos Git podem ser hospedados em plataformas como o GitHub.**

P.E.D

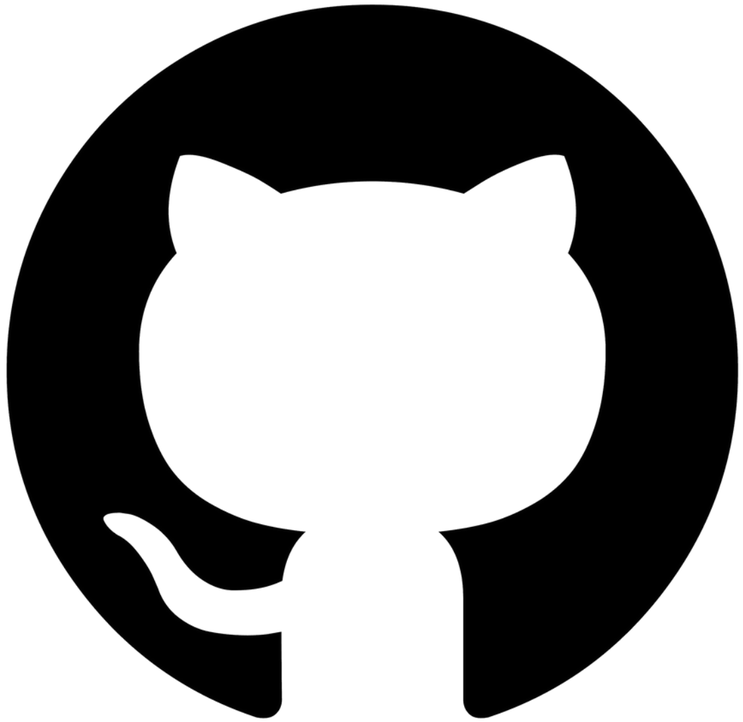
# GITHUB

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# O QUE É GITHUB?

- **O GitHub é uma plataforma de hospedagem de código fonte e colaboração baseada em Git.**
- Conhecido como Repositório remoto.
- Ele permite que os desenvolvedores compartilhem seus projetos de software.

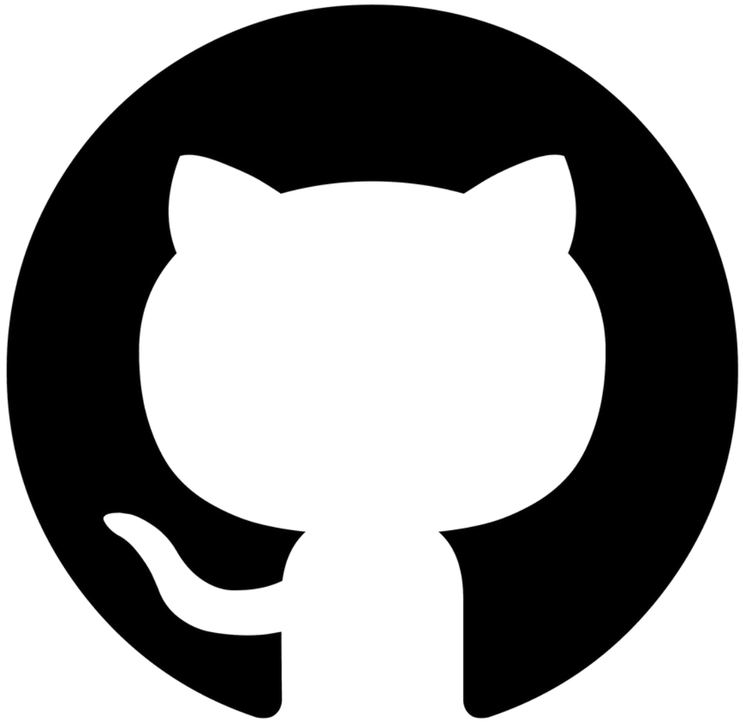




# O QUE É GITHUB?

- O GitHub é uma plataforma de hospedagem de código fonte e colaboração baseada em Git.
- **Conhecido como Repositório remoto.**
- Ele permite que os desenvolvedores compartilhem seus projetos de software.

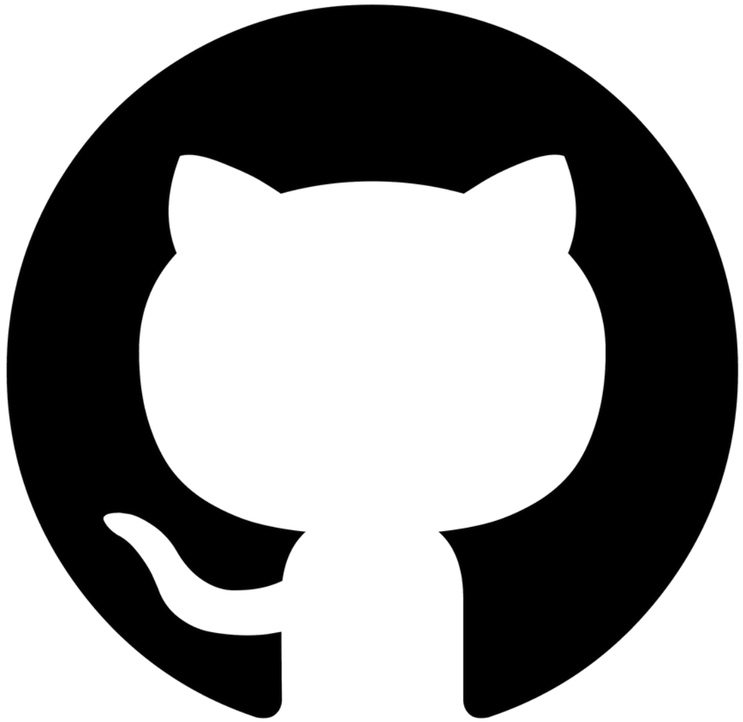




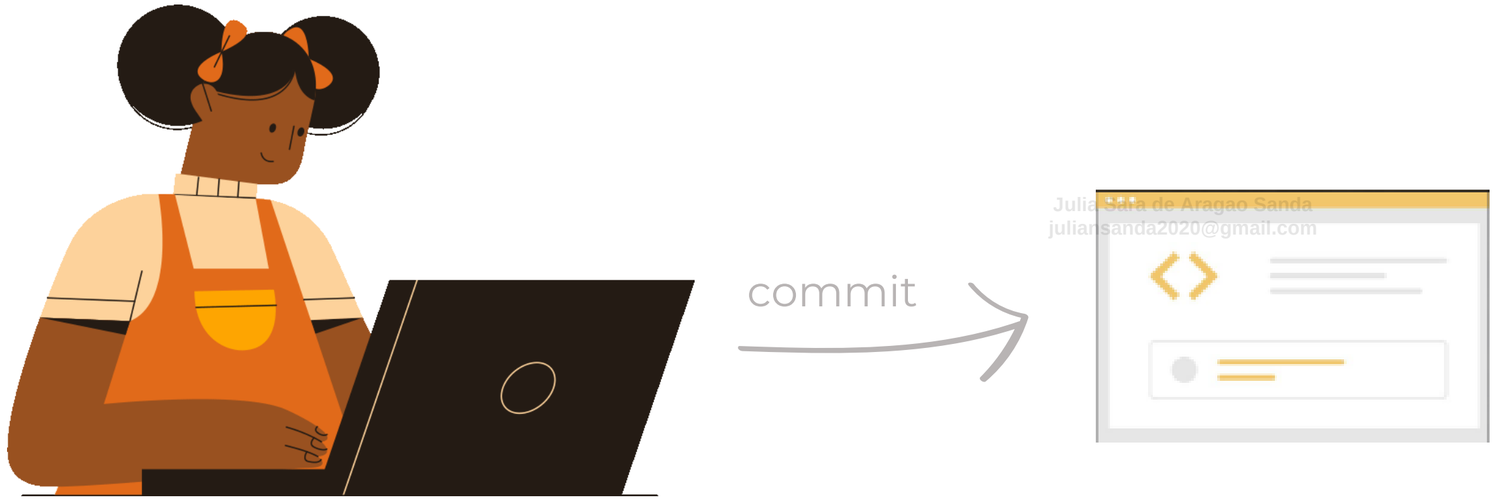
# O QUE É GITHUB?

- O GitHub é uma plataforma de hospedagem de código fonte e colaboração baseada em Git.
- Conhecido como Repositório remoto.
- **Ele permite que os desenvolvedores compartilhem seus projetos de software.**

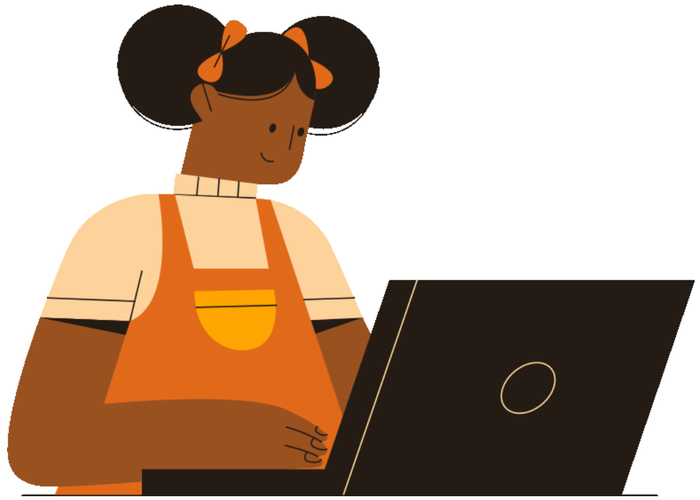




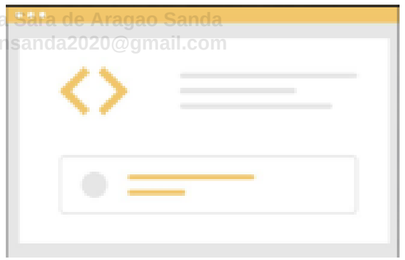
# GITHUB



# GITHUB

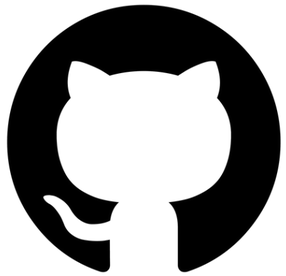


commit





push

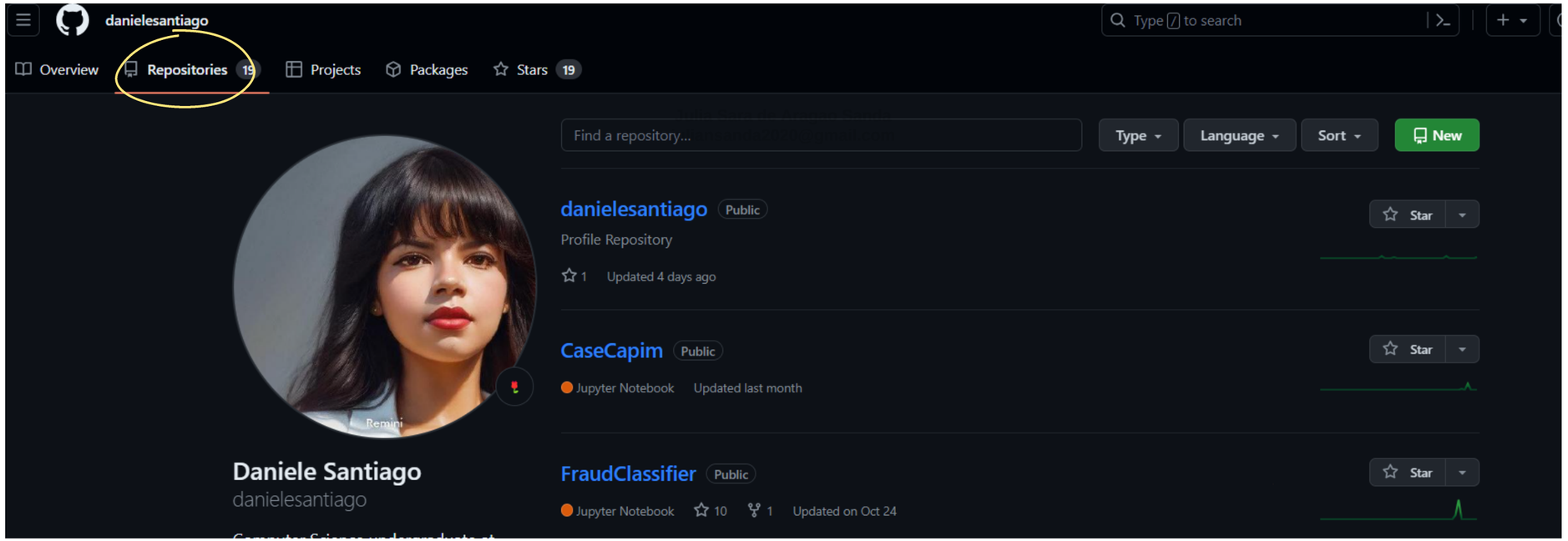


# REPOSITÓRIO

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# REPOSITÓRIO

Um **repositório** no GitHub é como uma pasta online onde você pode armazenar e gerenciar todos os arquivos relacionados a um projeto.



# BRANCH

Branch são versões paralelas do código-fonte em um repositório Git.

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# BRANCH

Branch são versões paralelas do código-fonte em um repositório Git.

Repositório

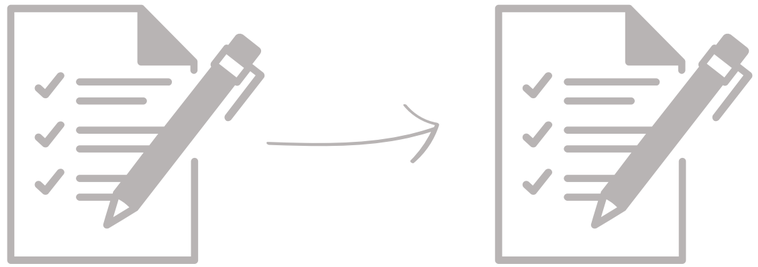
Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# BRANCH

Branch são versões paralelas do código-fonte em um repositório Git.

Repositório





Cópia

# BRANCH

Branch são versões paralelas do código-fonte em um repositório Git.

Repositório



Cópia

# BRANCH

**Pull Request**: Quando um desenvolvedor termina de trabalhar em um branch, ele pode abrir um "pull request" no GitHub. Isso é como uma **solicitação** para que as **mudanças** feitas no branch sejam **revisadas** e, se apropriadas, integradas ao branch principal.

**Revisão de Código**: Antes de um merge, outros membros da equipe geralmente revisam o código no pull request, fazem comentários e sugerem melhorias.



**Merge**: Após a revisão e aprovação do pull request, as mudanças podem ser incorporadas (ou "merged") no branch principal. Isso pode ser feito através de um botão de merge no GitHub.

**Merge Conflicts**: Às vezes, podem ocorrer conflitos de merge se o mesmo trecho de código foi modificado de maneiras diferentes nos dois branches. Estes conflitos precisam ser resolvidos manualmente antes que o merge possa ser completado.

# BOAS PRÁTICAS

- Um exemplo proeminente de boas práticas no desenvolvimento de software é o uso de "**Commits Convencionais**". Esta é uma especificação para adicionar significado legívell às mensagens de commit.

- Indicamos que seja seguido esse padrão. Suas regras, exemplos e especificações estão disponíveis em https://www.conventionalcommits.org/en/v1.0.0/#specification.

# EXERCÍCIO PRÁTICO



1. **Crie Sua Conta no GitHub**
   - Acesse github.com
   - Preencha as informações necessárias para criar sua conta
   - Confirme seu endereço de e-mail
2. **Explore o Painel Principal**
   - Familiarize-se com a interface de usuário
   - Verifique as sugestões de repositórios e usuários para seguir
3. **Descubra Repositórios**
   - Procure por repositórios interessantes
   - Observe como os projetos são estruturados

# USANDO GIT SEM TERMINAL

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# CONTEÚDO

1. **Objetivo**
   - Aprender a criar e gerenciar repositórios no GitHub via interface web.
   - Entender como editar e adicionar arquivos sem usar comandos de linha de comando.
2. **Criação de Repositórios na Web**
   - Como criar um novo repositório.
   - Adicionando uma descrição e configurando as opções do repositório.
   - Dicas para uma boa organização inicial.
3. **Edição e Adição de Arquivos**
   - Adicionar novos arquivos diretamente pela interface.
   - Editar arquivos existentes no navegador.
   - A importância de mensagens de commit claras.
4. **Branch**
   - Criar uma nova branch
   - Como fazer um pull request
   - Como fazer um merge e unir os dados

# EXERCÍCIO PRÁTICO



1. **Crie um novo repositório no GitHub.**
2. **Adicione arquivos diretamente pela interface web.**
3. **Crie uma branch “feature” e adicione mais arquivos.**
4. **Faça alteração no arquivo e crie um commit claro segundo as convenções.**
5. **Faça um pull request e um merge para a sua branch principal.**

# COMANDOS ESSENCIAIS DO GIT

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# COMANDOS ESSENCIAIS

- **git clone** // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote
- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- **git checkout “branch_name”** // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote
- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- **git fetch** // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote
- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- **git pull** // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote



- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- **git status** // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote



- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- **git add .** // Adiciona todas as suas alterações em um pacote
- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote



- **git commit -m “Comentário”** // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote
- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- **git push** // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote



- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- **git checkout -b “branch_name”** // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote



- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- **git merge “branch_name”** // Traz as alterações da branch_name para sua branch atual
- git branch -D “branch_atual” // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# COMANDOS ESSENCIAIS

- git clone // Copia o repositório
- git checkout “branch_name” // Troca de branch
- git fetch // Atualiza suas referências locais em referências remotas
- git pull // Atualiza sua branch local com a versão atual do git
- git status // Apresenta o status da branch local contra a do git
- git add . // Adiciona todas as suas alterações em um pacote
- git commit -m “Comentário” // Comenta as alterações que serão enviadas
- git push // Envia as alterações adicionadas e comentadas
- git checkout -b “branch_name” // Clona sua branch atual com o nome branch_name
- git merge “branch_name” // Traz as alterações da branch_name para sua branch atual
- **git branch -D “branch_atual”** // Apaga sua branch localmente
  *obs: cuidado para não apagar a branch errada*

# USANDO GIT NO TERMINAL

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# PASSO A PASSO PARA UTILIZAR O GIT NO TERMINAL

1. **Realizar a instalação do GIT**
2. Abra seu terminal (cmd) e coloque suas credenciais:
   git config --global user.name “Seu nome”
   git config - - global user.email “Email do git”
3. Tenha acesso ao repositório do git que você quer clonar.

# PASSO A PASSO PARA UTILIZAR O GIT NO TERMINAL

1. Realizar a instalação do GIT
2. **Abra seu terminal (cmd) e coloque suas credenciais:**

git config - -global user.name “Seu nome”

git config - - global user.email “Email do git”

3. Tenha acesso ao repositório do git que você quer clonar.

# PASSO A PASSO PARA UTILIZAR O GIT NO TERMINAL

1. Realizar a instalação do GIT
2. Abra seu terminal (cmd) e coloque suas credenciais:
git config --global user.name “Seu nome”
git config -- global user.email “Email do git”
3. **Tenha acesso ao repositório do git que você quer clonar.**

# EXERCÍCIO PRÁTICO



1. **Clone um repositório utilizando o terminal.**
2. **Adicione arquivos, faça um commit e mande ao repositório remoto.**
3. **Crie uma branch “feature” e adicione mais arquivos.**
4. **Faça um pull request e um merge da branch “feature” para a sua branch principal.**

# GITHUB NO MUNDO CORPORATIVO

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# WORKFLOW

- **No mundo corporativo é utilizado um workflow para a organização de branches.**
- Ele é responsável por ditar com as branches são nomeadas, organizadas e quais movimentações devem ser feitas.
- Ao utilizarmos um workflow, conseguimos assegurar a segurança e integridade dos projetos, promovendo tanto o versionamento quanto uma maior facilidade de manipulação do histórico de commits.

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# WORKFLOW

- No mundo corporativo é utilizado um workflow para a organização de branches.
- **Ele é responsável por ditar com as branches são nomeadas, organizadas e quais movimentações devem ser feitas.**
- Ao utilizarmos um workflow, conseguimos assegurar a segurança e integridade dos projetos, promovendo tanto o versionamento quanto uma maior facilidade de manipulação do histórico de commits.

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# WORKFLOW

- No mundo corporativo é utilizado um workflow para a organização de branches.
- Ele é responsável por ditar com as branches são nomeadas, organizadas e quais movimentações devem ser feitas.
- **Ao utilizarmos um workflow, conseguimos assegurar a segurança e integridade dos projetos, promovendo tanto o versionamento quanto uma maior facilidade de manipulação do histórico de commits.**

Julia Sara de Aragao Sanda
juliansanda2020@gmail.com

# WORKFLOW

- No mundo corporativo é utilizado um workflow para a organização de branches.
- Ele é responsável por ditar com as branches são nomeadas, organizadas e quais movimentações devem ser feitas.
- Ao utilizarmos um workflow, conseguimos assegurar a segurança e integridade dos projetos, promovendo tanto o versionamento quanto uma maior facilidade de manipulação do histórico de commits.



Branches Principais

**Master**

Develop

Homolog

Branches Secundárias

Feature

Delivery

# WORKFLOW

- No mundo corporativo é utilizado um workflow para a organização de branches.
- Ele é responsável por ditar com as branches são nomeadas, organizadas e quais movimentações devem ser feitas.
- Ao utilizarmos um workflow, conseguimos assegurar a segurança e integridade dos projetos, promovendo tanto o versionamento quanto uma maior facilidade de manipulação do histórico de commits.



Branches Principais

Master

**Develop**

Homolog

Branches Secundárias

Feature

Delivery

# WORKFLOW

- No mundo corporativo é utilizado um workflow para a organização de branches.
- Ele é responsável por ditar com as branches são nomeadas, organizadas e quais movimentações devem ser feitas.
- Ao utilizarmos um workflow, conseguimos assegurar a segurança e integridade dos projetos, promovendo tanto o versionamento quanto uma maior facilidade de manipulação do histórico de commits.



Branches Principais

Master

Develop

**Homolog**

Branches Secundárias

Feature

Delivery

# WORKFLOW

- No mundo corporativo é utilizado um workflow para a organização de branches.
- Ele é responsável por ditar com as branches são nomeadas, organizadas e quais movimentações devem ser feitas.
- Ao utilizarmos um workflow, conseguimos assegurar a segurança e integridade dos projetos, promovendo tanto o versionamento quanto uma maior facilidade de manipulação do histórico de commits.



Branches Principais

Master

Develop

Homolog

Branches Secundárias

**Feature**

Delivery

# WORKFLOW

- No mundo corporativo é utilizado um workflow para a organização de branches.
- Ele é responsável por ditar com as branches são nomeadas, organizadas e quais movimentações devem ser feitas.
- Ao utilizarmos um workflow, conseguimos assegurar a segurança e integridade dos projetos, promovendo tanto o versionamento quanto uma maior facilidade de manipulação do histórico de commits.



Branches Principais

Master

Develop

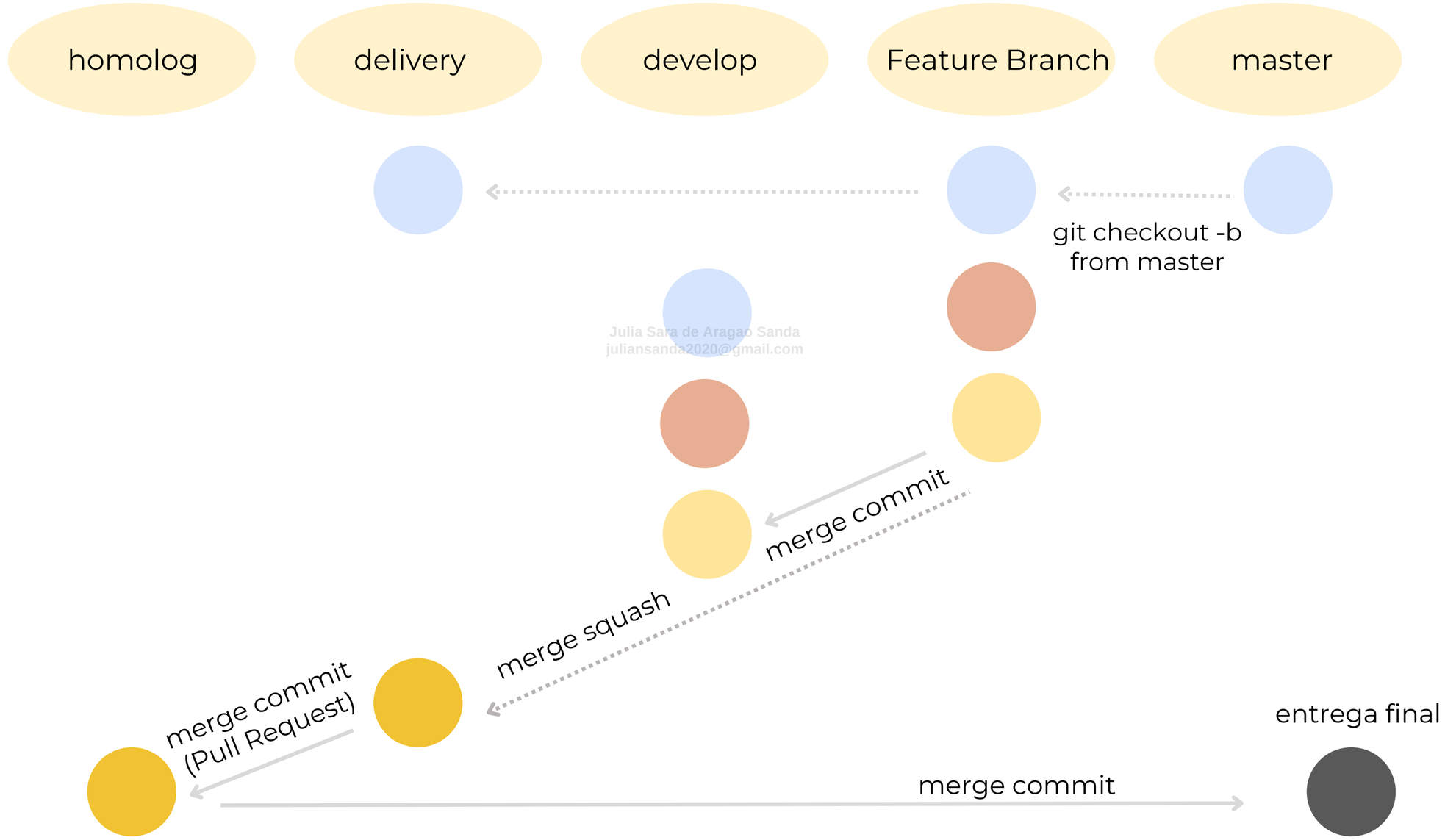
Homolog

Branches Secundárias

Feature

**Delivery**

# WORKFLOW



# OBRIGADA!