

000  
010  
110  
10  
0  
1  
0

MINICURSO



# Descomplicando o



# git

PARTE 1 | 2019  
SACIS

# Apresentação



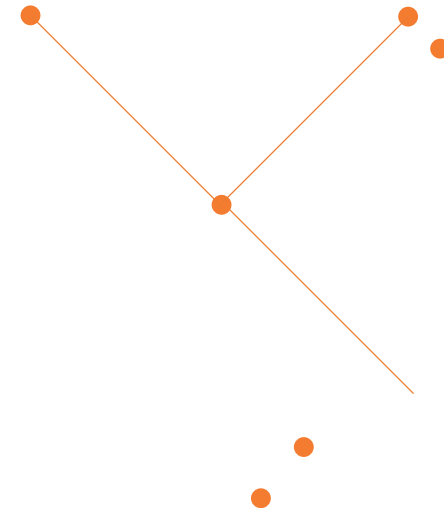
Nome: Nádia Oliveira  
Cidade: São Sebastião do Rio Preto - MG  
Faculdade: Sistemas de Informação  
O que eu fiz até aqui?

0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# ALERTA !



O conteúdo deste  
minicurso pode afetar  
sua **ZONA DE CONFORTO**



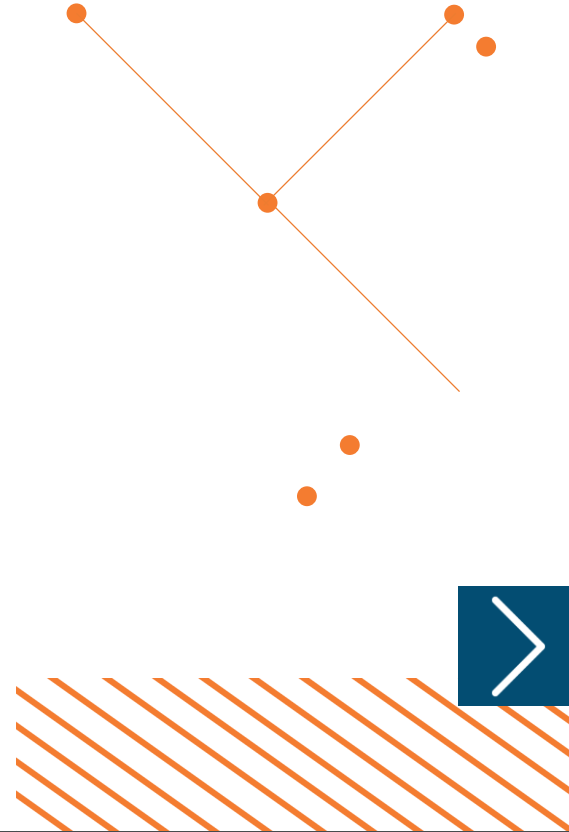
0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Situação problema —

Como desenvolver em equipe

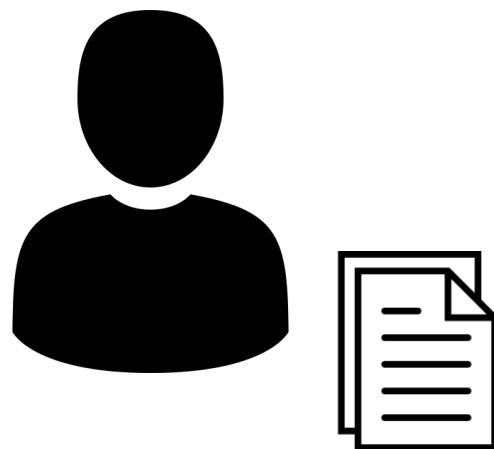


E se você precisar voltar a uma versão anterior do seu código



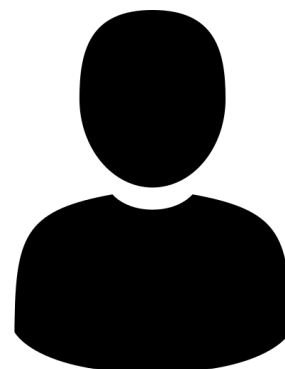
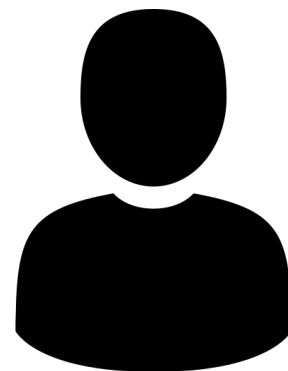
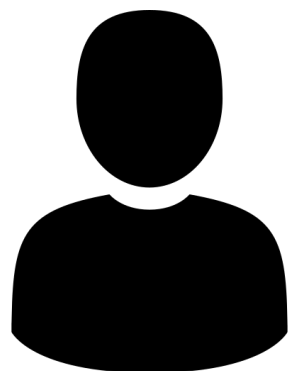
0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Situação problema —

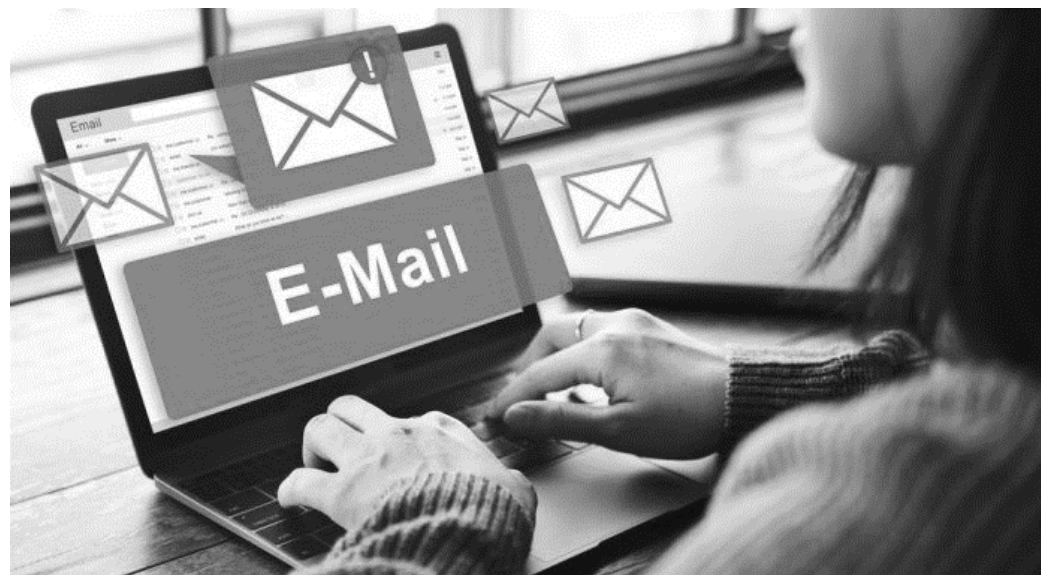
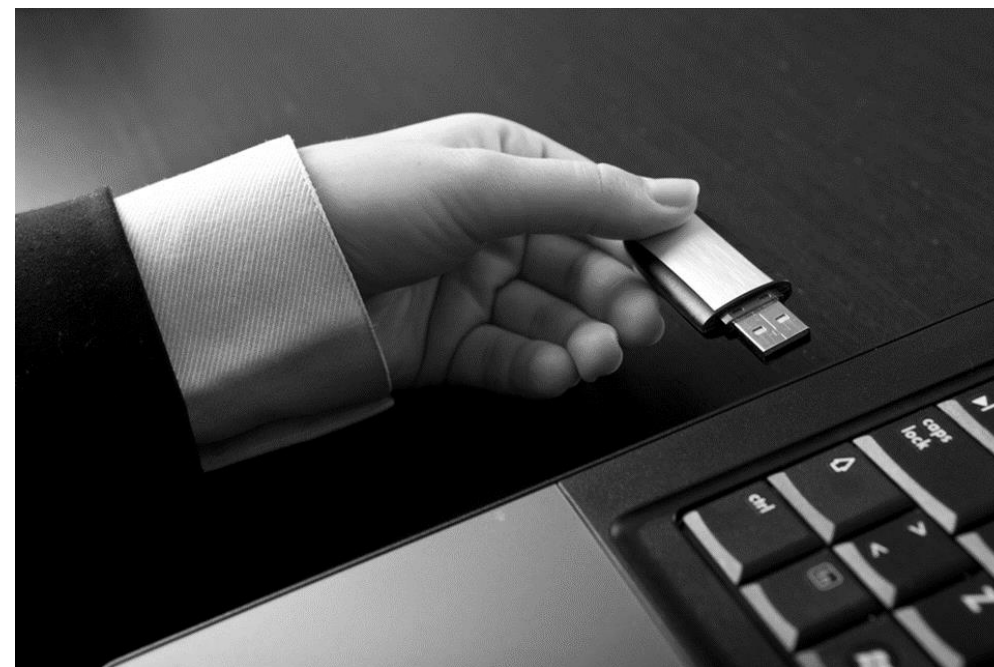


000  
010  
110  
10  
0  
1  
0

# Situação problema —

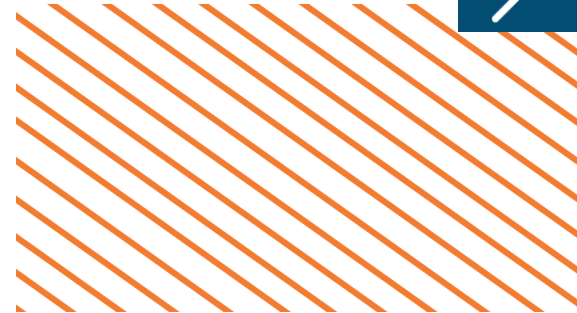
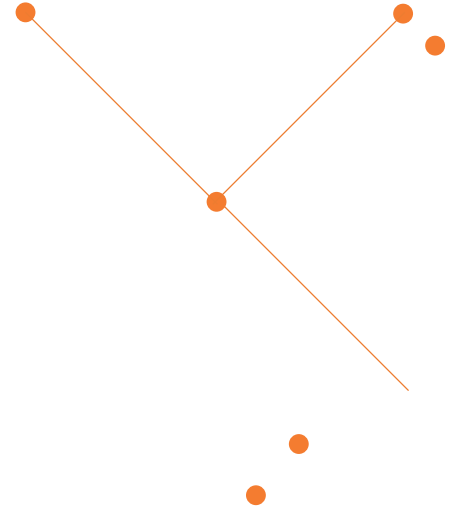
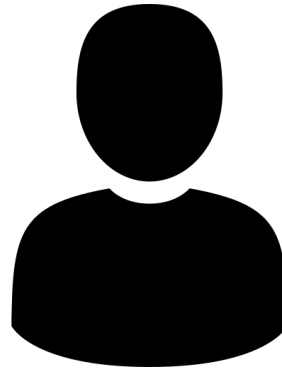
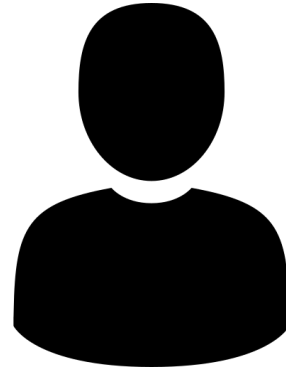
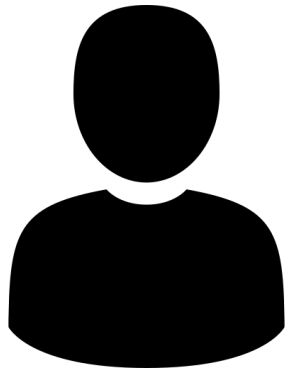


# SOLUÇÃO



0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

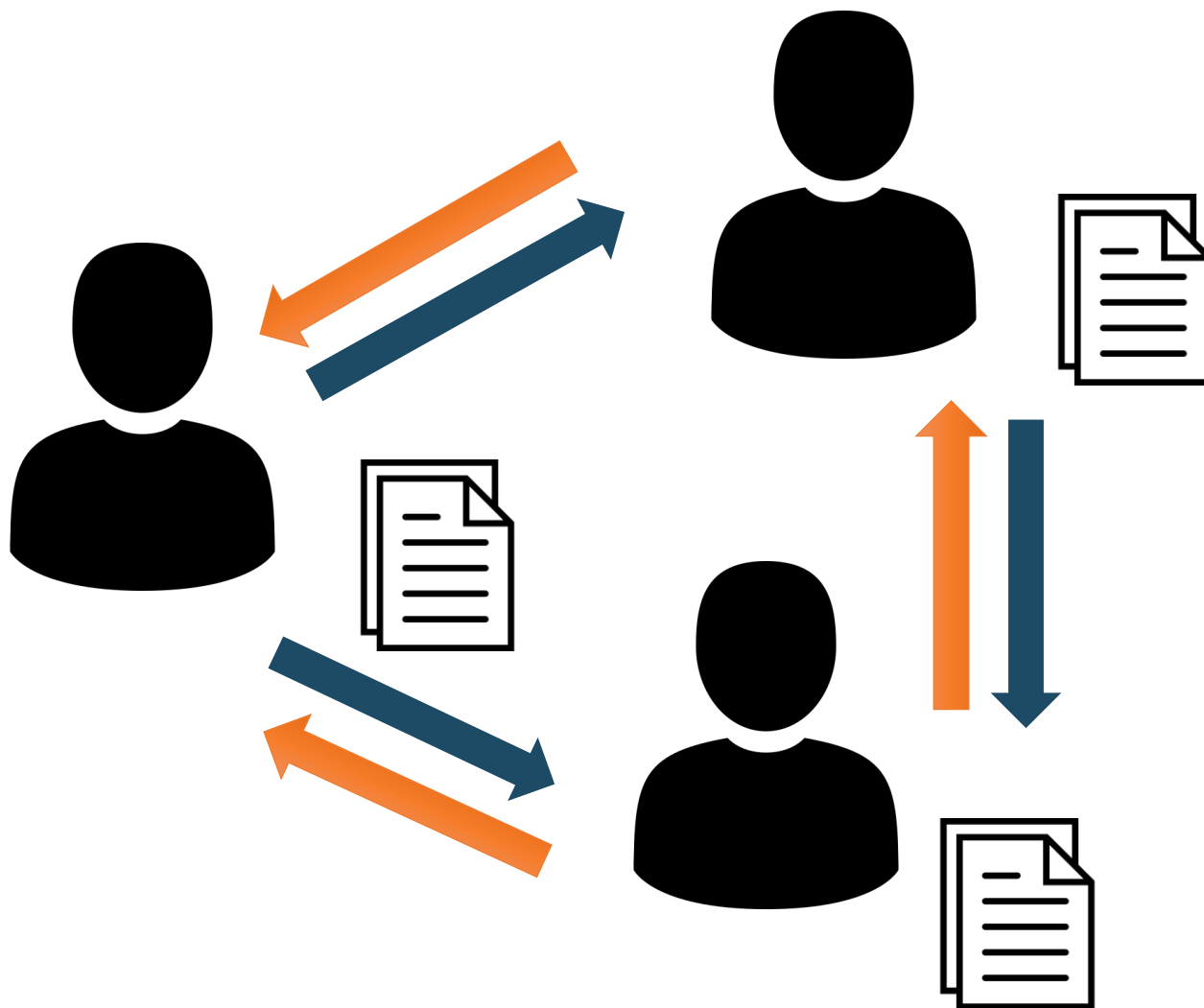
# Situação problema —





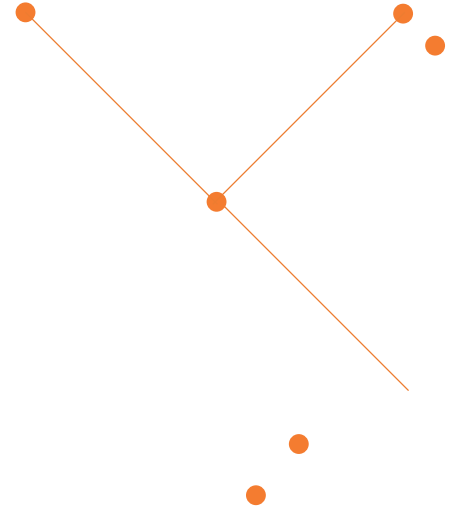
000  
010  
110  
10  
0  
1  
0

# Situação problema —



# Problemas da solução anterior

- Excesso de arquivos
  - Arquivos duplicados
  - Dependência

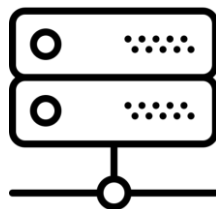
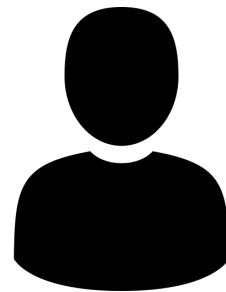
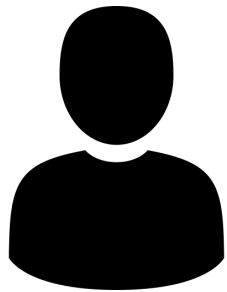
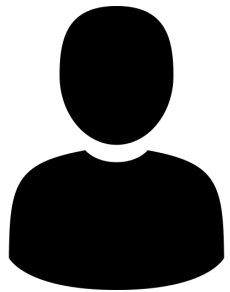


Solução  
eficaz:

**sistema de controle de versão**

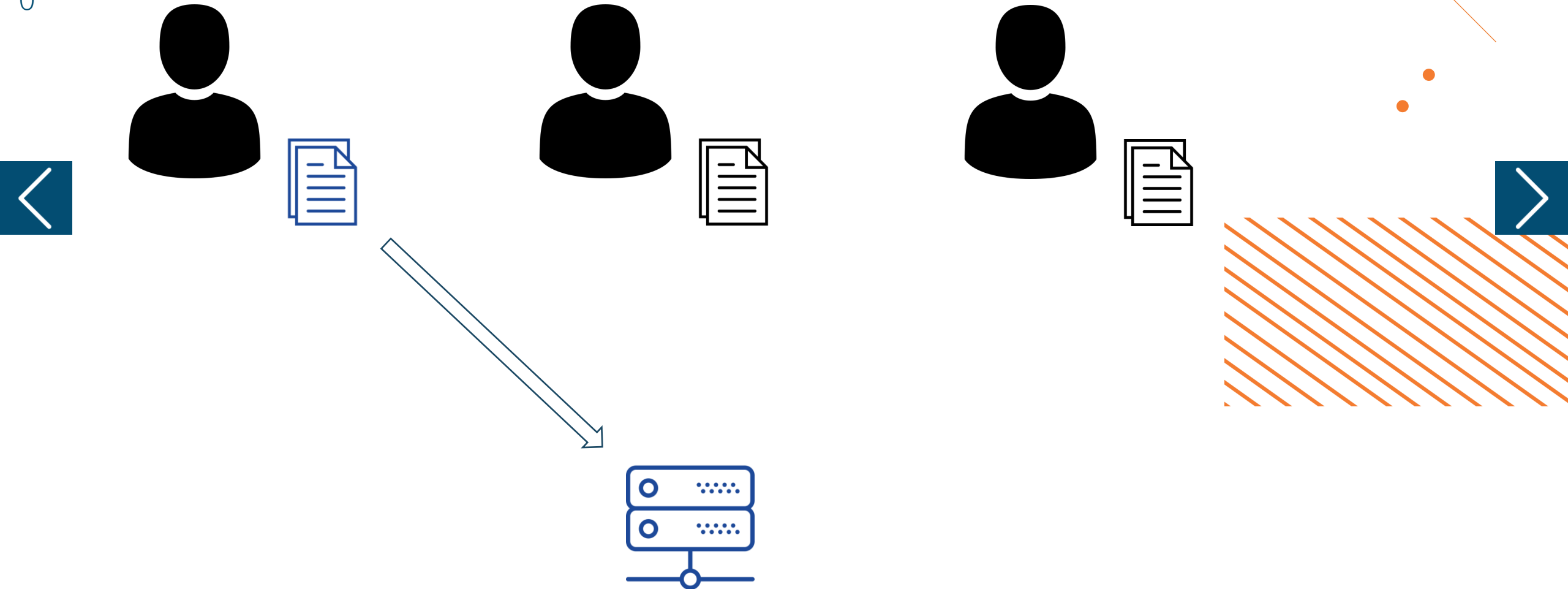
0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Sistema de Controle de Versão



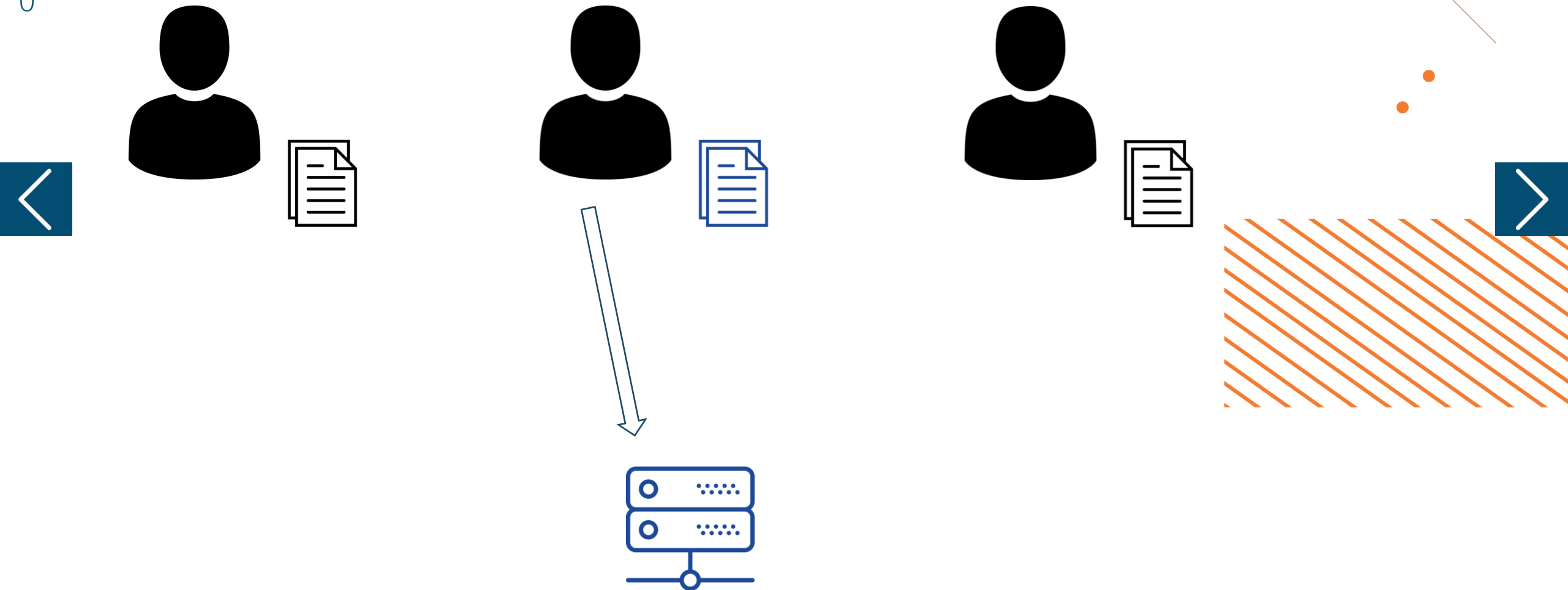
0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Sistema de Controle de Versão



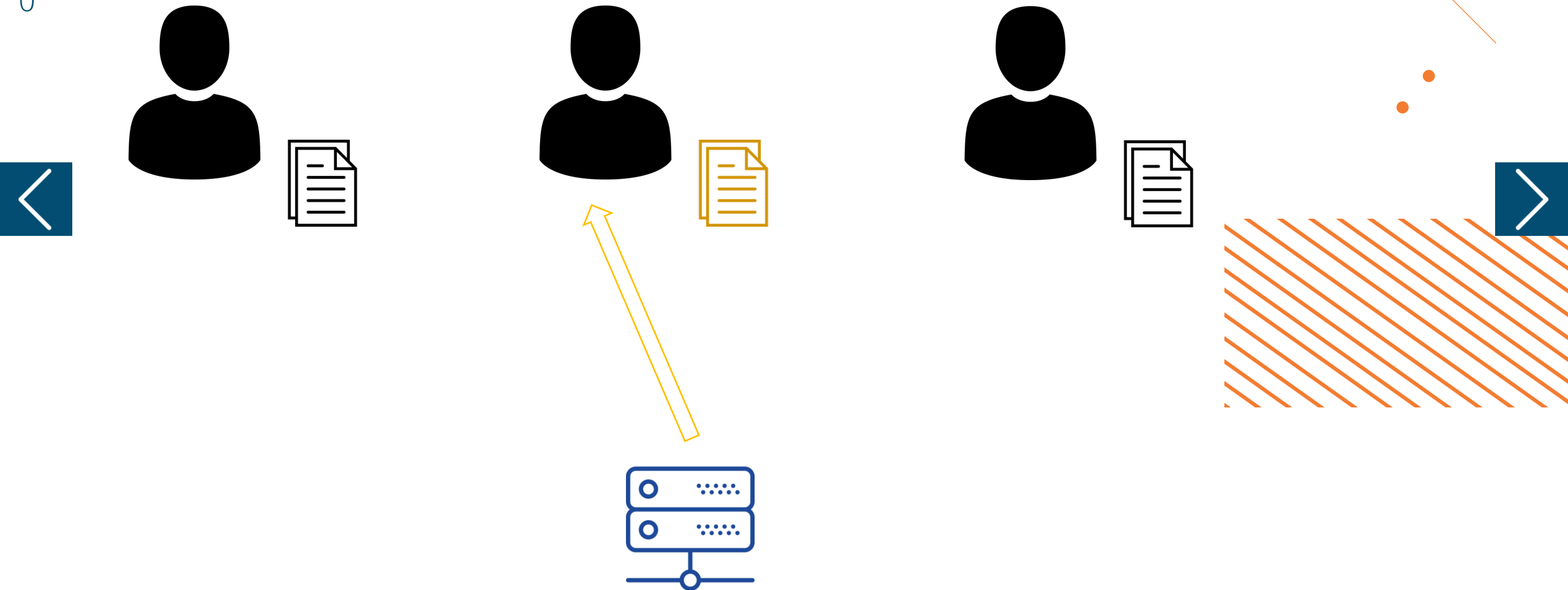
0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Sistema de Controle de Versão



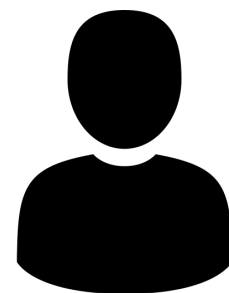
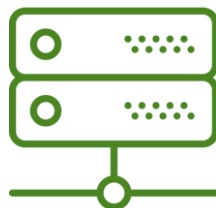
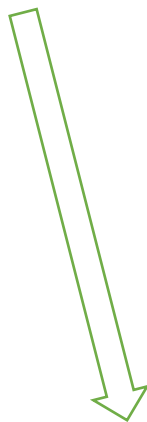
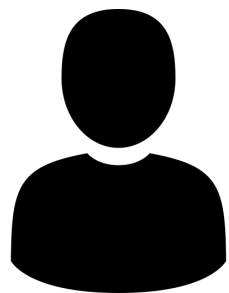
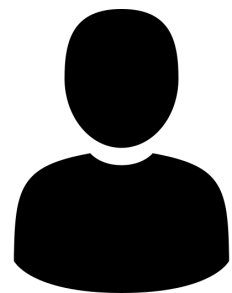
000  
010  
110  
10  
0  
1  
0

# Sistema de Controle de Versão



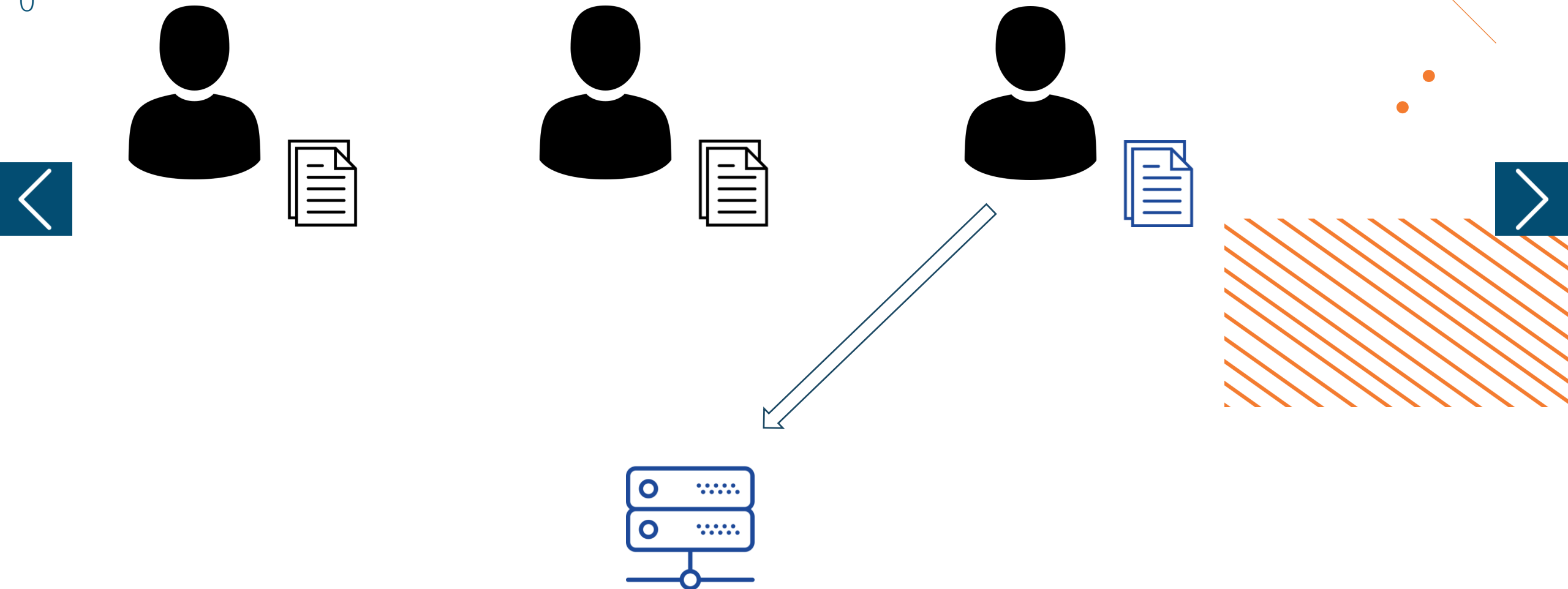
0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Sistema de Controle de Versão



000  
010  
110  
10  
0  
1  
0

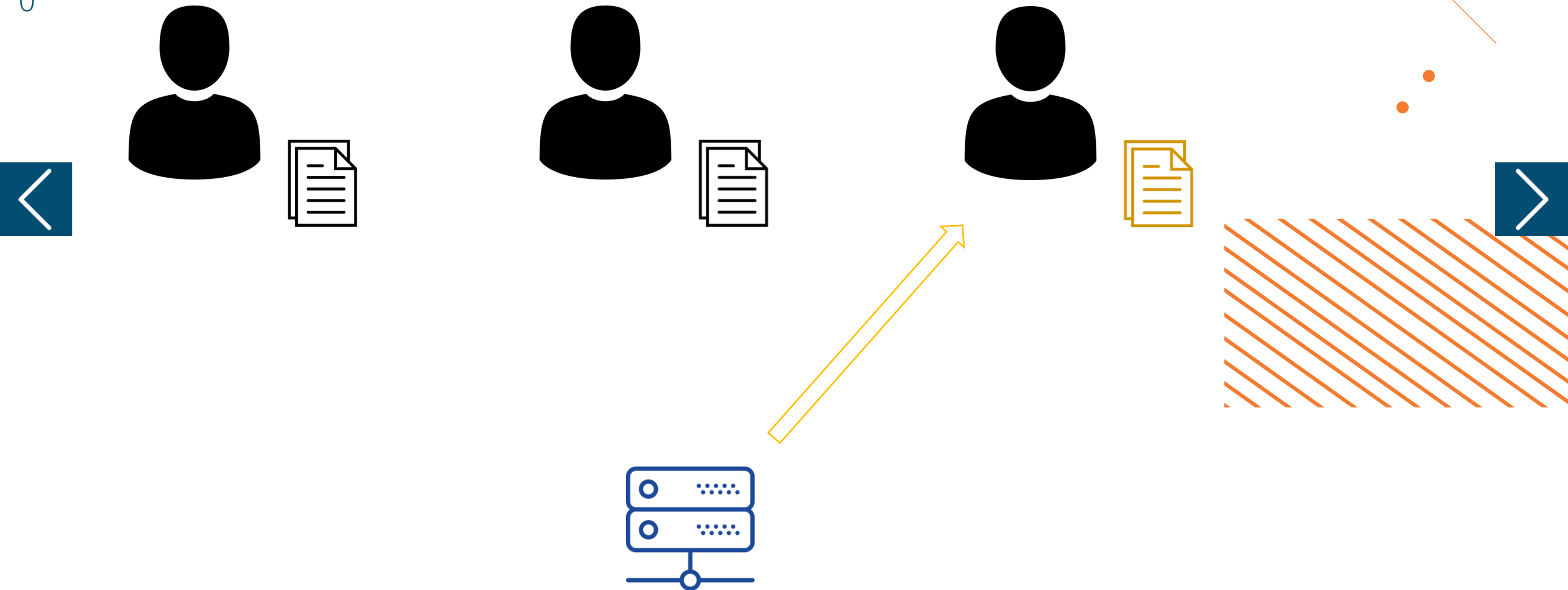
# Sistema de Controle de Versão





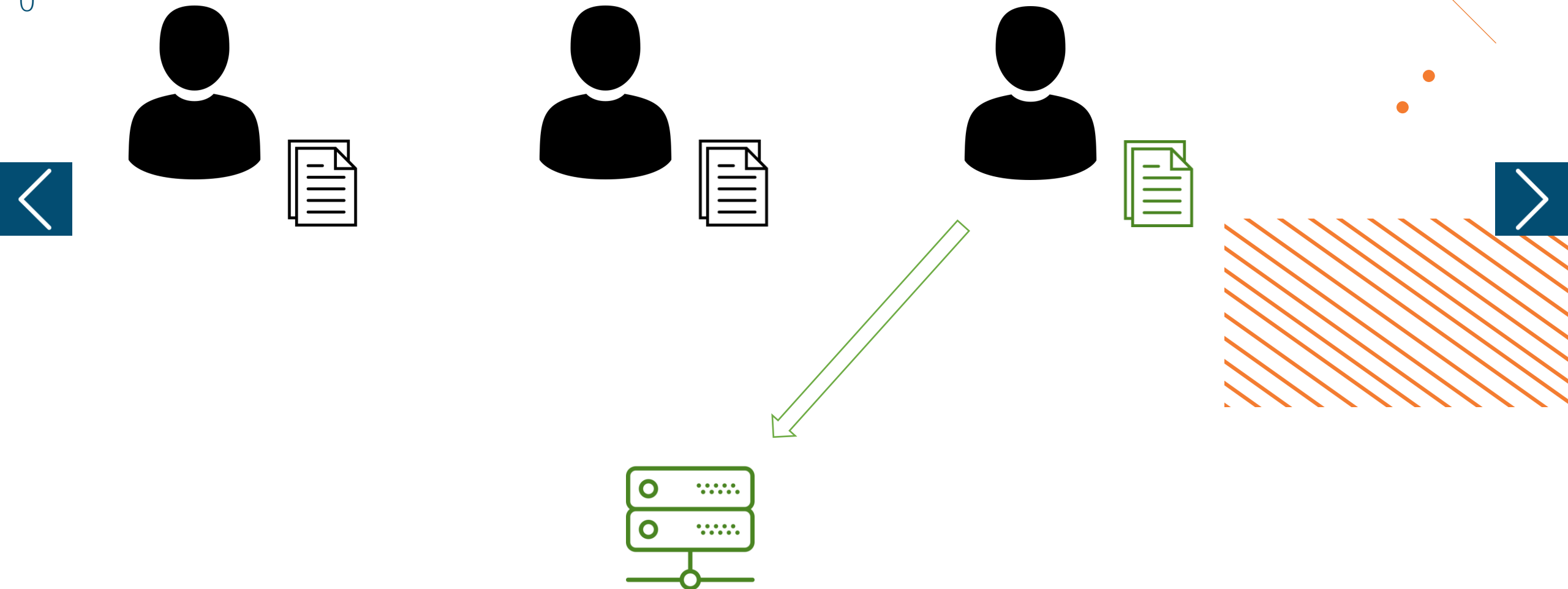
000  
010  
110  
10  
0  
1  
0

# Sistema de Controle de Versão




000  
010  
110  
10  
0  
1  
0


# Sistema de Controle de Versão





0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0


# Sistema de Controle de Versão


 ACABOOOOOOOOOOUUUU  
C Source File  
0 bytes

 JogoDaVelhaV2  
C Source File  
0 bytes

 Agoravaaaaaiii  
C Source File  
0 bytes

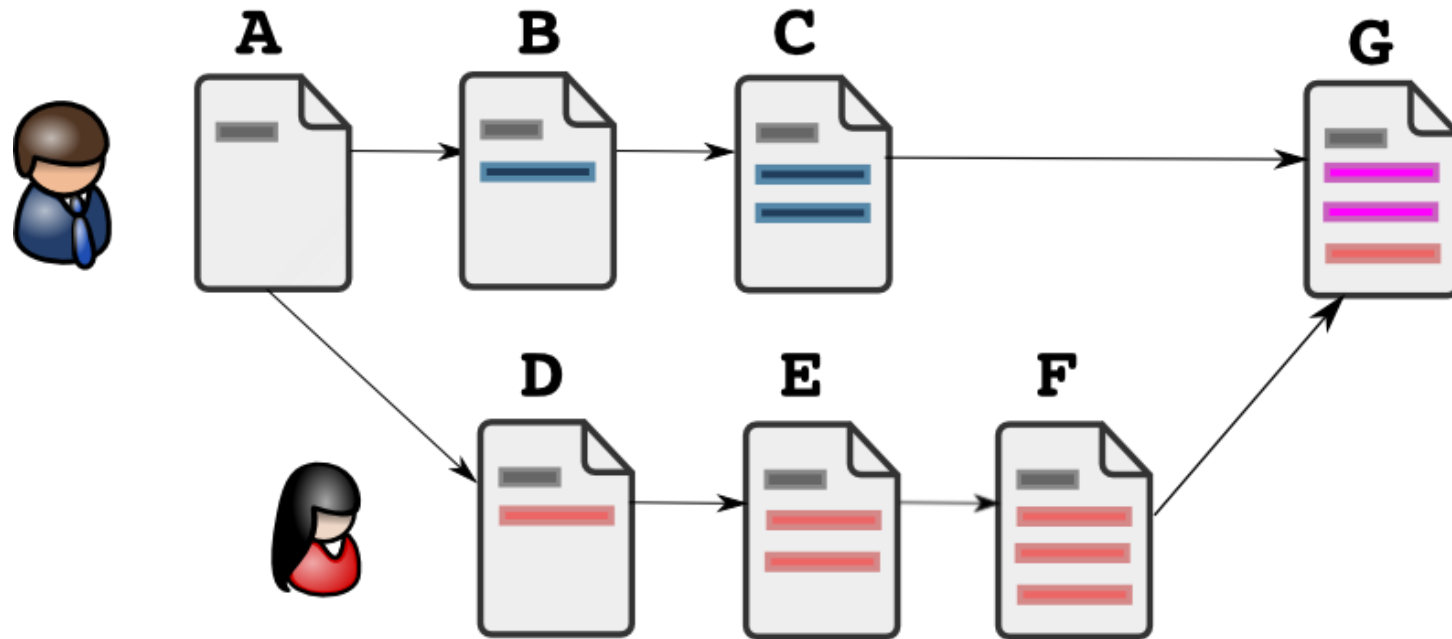
 QuaseProntoooo  
C Source File  
0 bytes

 JogoDaVelhaV1  
C Source File  
0 bytes

 TaComErro  
C Source File  
0 bytes

# O que é um sistema de controle de versão?

Qualquer sistema que possa prover rastreamento e controle sobre as mudanças de documentos



# Alguns sistemas de controle de versão

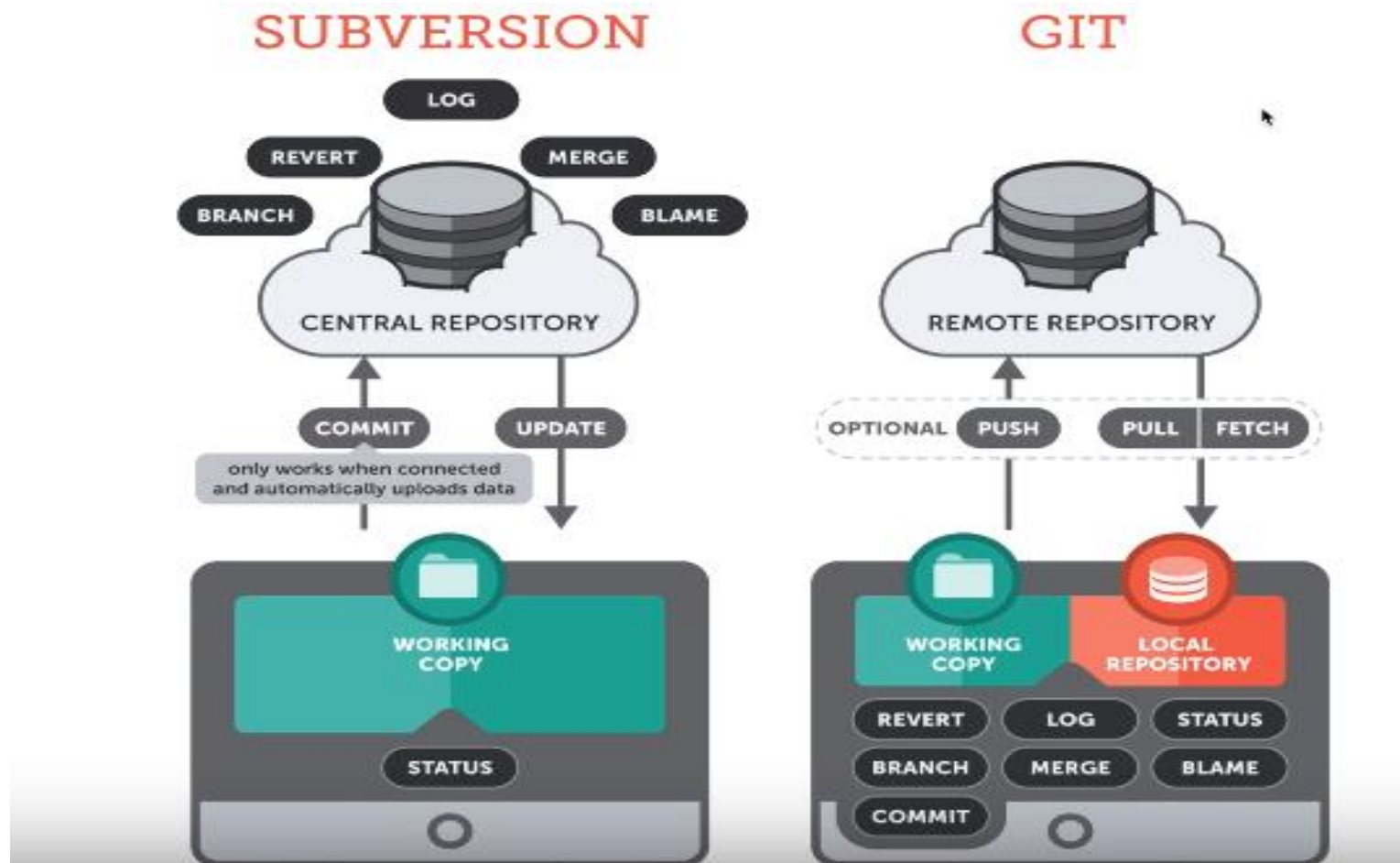


**CVS**



...

# Tipos de sistemas de controle de versão

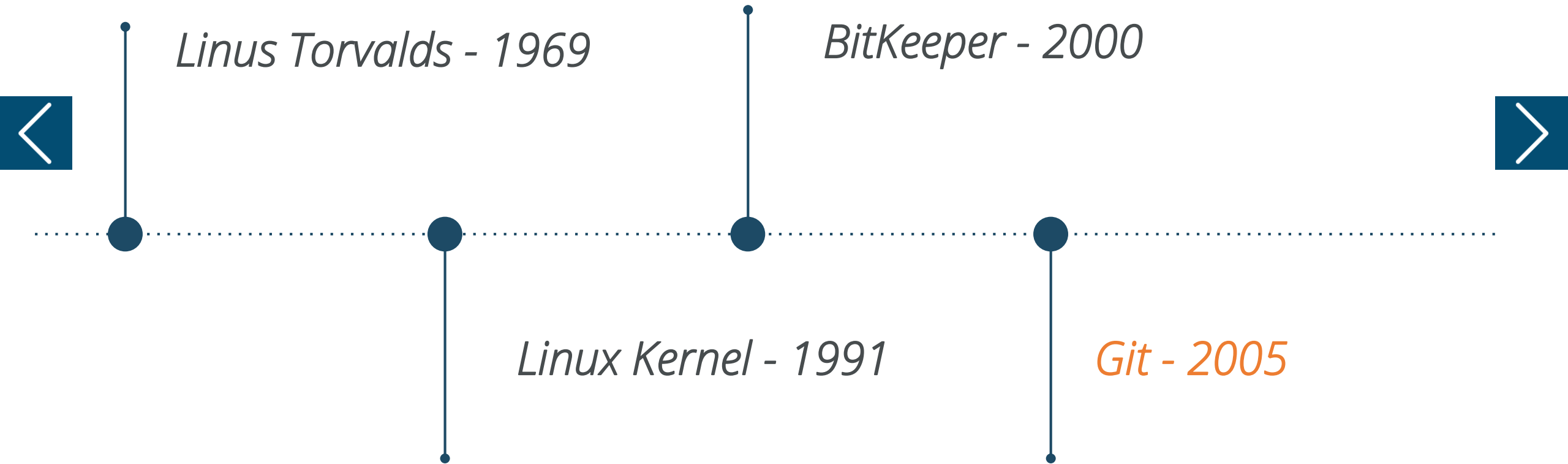
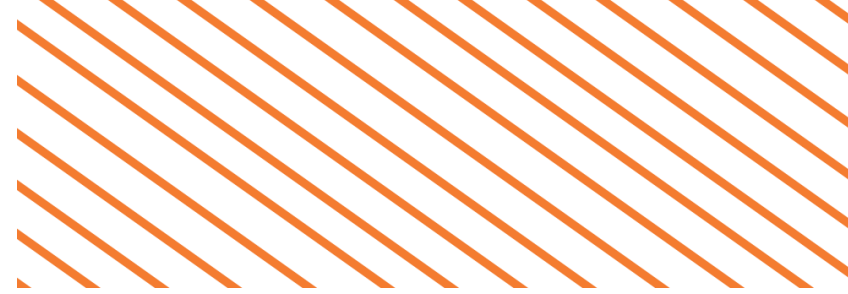


# Nosso foco



git

# História do git






# Linus Torvalds





# O que é o git?

“ É um **sistema** de gerenciamento de arquivos com **controle de versão** distribuído. ”

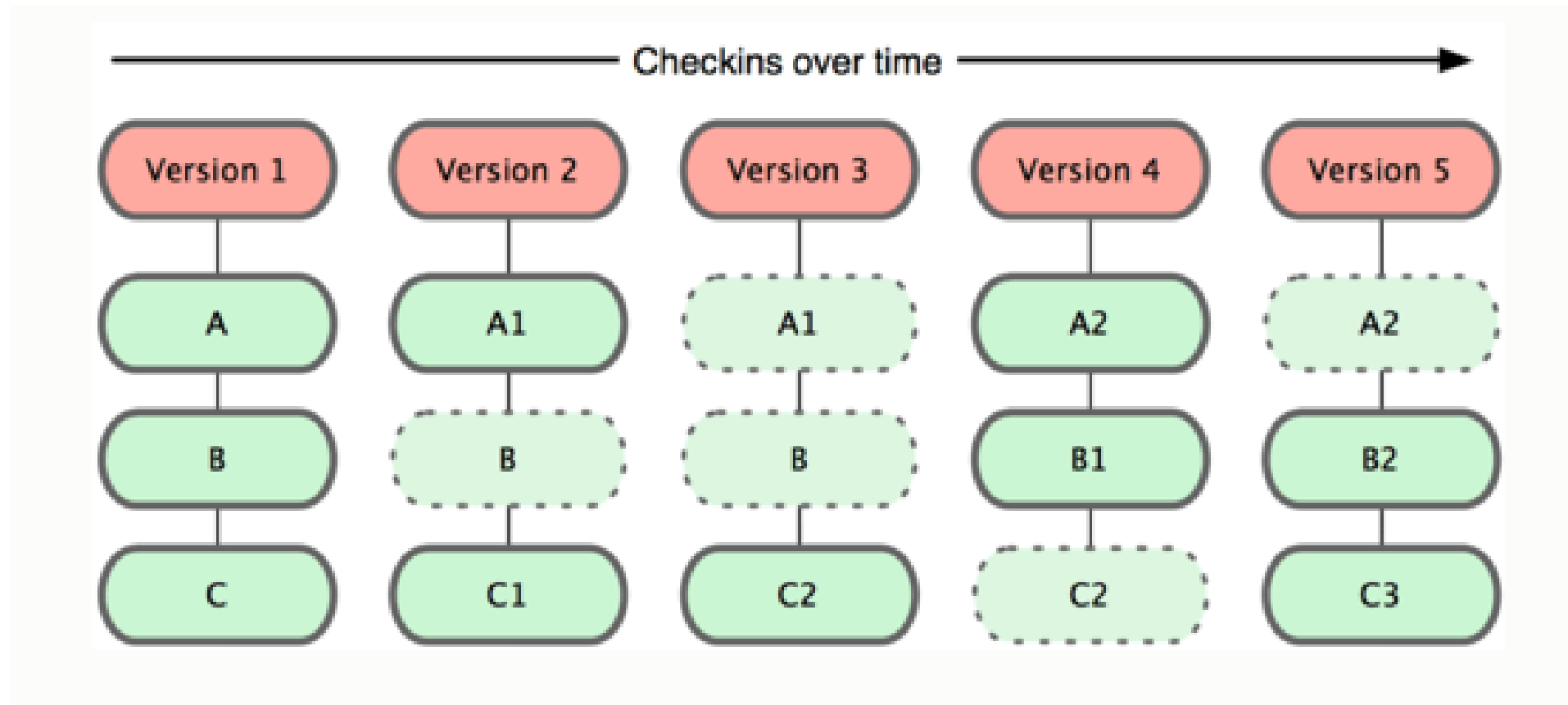
# Por que o git se difere dos demais scv?



A maior diferença está na forma que o Git trata os dados. Não é o que ele faz, mas **COMO** ele faz.



# Snapshots, e não diferenças



# Por que VOCÊ deve usar um SCV?

- ✓ Confiabilidade
- ✓ Produtividade
- ✓ História do código ao longo do tempo
- ✓ Identifica o autor da mudança no código para poder: **XINGÁ-LO**

000  
010  
110  
10  
0  
1  
0

MINICURSO



# Descomplicando o



# git

PARTE 2 | 2019  
SACIS

# Instalando o Git



1. Acesse: ***<https://git-scm.com/>***
2. Clique em ***Download for Mac***
3. Extraia os arquivos e execute o .pkg
4. Após a conclusão da instalação, abra o terminal.
5. Digite: ***git --version*** e deverá aparecer a versão do git

# Instalando o Git



1. Abra o ***terminal***
2. Caso esteja em uma versão baseada no ***Debian***, digite: ***sudo apt-get install git***
3. Caso esteja no ***Fedora***, digite: ***yum install git-core***
4. Digite: ***git --version*** e deverá aparecer a versão do git



# Instalando o Git



1. Acesse: ***<https://git-scm.com/>***
2. Clique em ***Download for Windows***
3. Execute o .exe
4. Após a conclusão da instalação, abra o ***git bash***
5. Digite: ***git --version*** e deverá aparecer a versão do git

# git --version

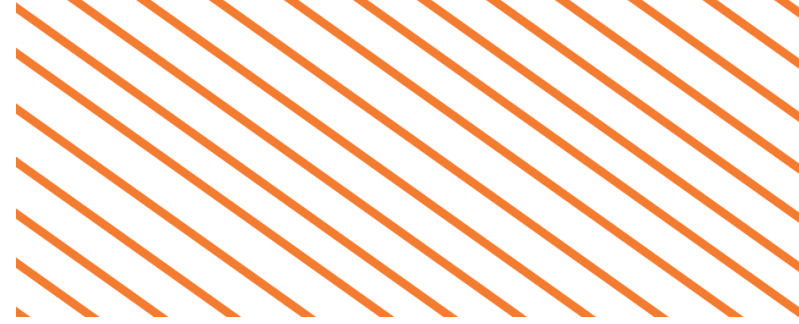
Resultado esperado



```
Nádia Oliveira@DESKTOP-IQ008BE MINGW64 ~  
$ git --version  
git version 2.20.1.windows.1
```

# git --version

O que faz?



Permite ver se o Git está instalado na máquina bem como sua versão de instalação.



000  
010  
110  
10  
0  
1  
0

MINICURSO



# Descomplicando o git

PRÁTICA | 2019  
SACIS

# Conceitos que VOCÊ precisa saber

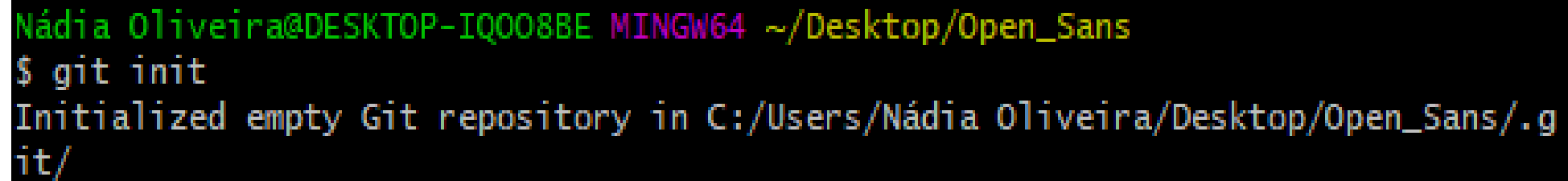
- Repositório: local onde fica todos os arquivos do projeto, inclusive os históricos e versões.

# Criando um diretório

1. Crie uma *pasta* e a nomeie como preferir
2. Abra o DEV++
3. Crie um arquivo *“projeto.c”* e salve na pasta que criou
4. Vá até a pasta e com o botão esquerdo selecione *Git Bash Here*
5. Digite *git init*

# git init

Resultado esperado



```
Nádia Oliveira@DESKTOP-IQ008BE MINGW64 ~/Desktop/Open_Sans
$ git init
Initialized empty Git repository in C:/Users/Nádia Oliveira/Desktop/Open_Sans/.git/
```

# git init

O que faz?



Permite inicializar um repositório do git, a partir desse comando o git passa a monitorar os arquivos presentes na pasta.





# Configurações

- Configurando o nome de usuário

***git config user.name "seu nome"***

- Configurando o e-mail

***git config user.email "meuemail@dominio.com"***



# Como saber quais arquivos o git está monitorando ?

1. Digite no bash: *git ls-files*

Este comando nos retorna todos os arquivos que o git está monitorando

0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Codificando o arquivo projeto.c

- 
- Vamos codificar uma calculadora que permite somar dois números
- 

# Alguns comandos no terminal

- GIT STATUS

Este comando serve para checar o estado atual do repositório.



# Adicionando o arquivo ao git

1. Digite no bash: *git add projeto.c*

Este comando diz para o GIT que queremos que ele monitore este arquivo.

# Enviando comentário do que foi feito

1. Digite: *git commit -m "mensagem"*

Este comando serve para que possamos identificar o que foi feito através de uma mensagem simples.

# Mais comandos... `git log`



Este comando nos permite ver todos os commits feitos ao longo do repositório



0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Codificando o arquivo projeto.c

- 
- Vamos adicionar a função de subtração na calculadora
- 



0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# Codificando o arquivo projeto.c

- 
- Vamos adicionar a função de multiplicação na calculadora
- 

0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

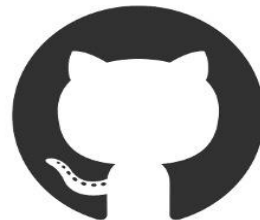
# Codificando o arquivo projeto.c

- 
- Vamos adicionar a função de divisão na calculadora
- 

# E se eu quiser disponibilizar meu código na rede para trabalhar em equipe ou para tê-lo guardado na nuvem ?



**Bitbucket**



**GitLab**

# Nosso foco



# GitHub

# Criando uma conta

- 
- Acesse: ***https://github.com/***
- 

# Criando o repositório no github

- 
- 
1. *New repository*
  2. *Nome do repositório, por exemplo: "Minicurso Git"*



# Enviando repositório local para repositório remoto

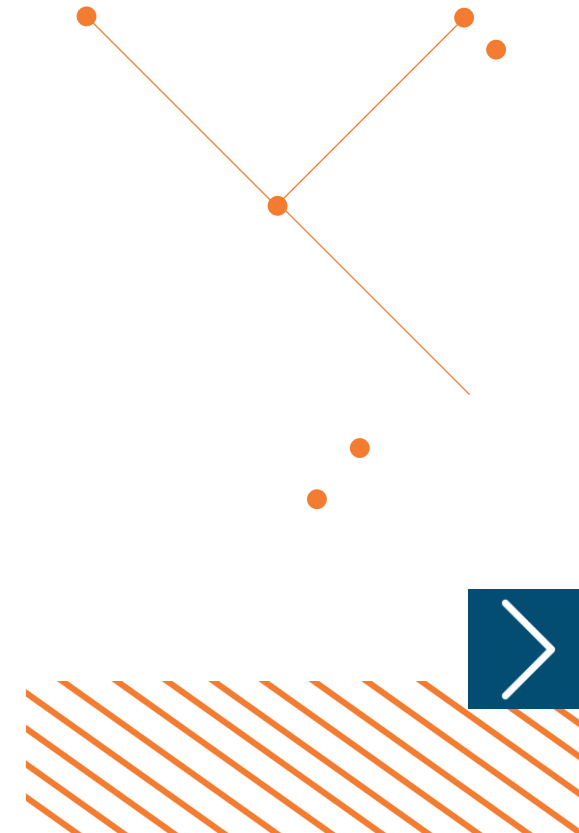
- git remote
- git remote add origin [url]
- git push -u origin master

O comando **“git remote”** mostra quais repositórios remotos a máquina tem, já o **“git remote add origin[url]”** adiciona um novo repositório remoto na máquina. Para mandar as alterações para o repositório remoto, usamos o comando o **“git push origin master”**

0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# O QUE VIMOS ATÉ AQUI

- O que é SCV
- O que é git
- Como o git trabalha
- História do git
- Como monitorar arquivos com git
- O que é github
- Enviando repositório local para repositório remoto





000  
010  
110  
10  
0  
1  
0

MINICURSO



Descom  
plicando o




git

COFFEE - BREAK | 2019  
SACIS

# E se eu quiser ver o que foi alterado em cada commit?

- `git whatchanged -p`



Este comando mostra detalhadamente o que foi alterado em cada commit



# Tratando exceções no projeto.c

- Vamos emitir uma mensagem de erro caso o usuário digite o divisor seja zero, commitar e enviar nossas modificações para o github



# Como voltar a uma versão anterior do projeto?



**Só execute** os procedimentos a seguir caso tenha **CERTEZA** do que irá fazer.



# Como voltar a uma versão anterior do projeto?

- git log (copie a hash da versão que deseja voltar)
- git checkout <hash da version>
- git checkout master
- git reset --hard <hash da version>
- git push origin master -f

Executando o comando git reset --hard, os commits posteriores são descartados

# Escrevi um commit errado e agora?

- Se foi o último commit que escreveu, basta digitar:

1. `git commit --amend`

- Para commits anteriores:

1. `git rebase -i HEAD~5` // Mostra a lista dos 5 últimos commits
2. Onde você quer alterar a mensagem troque de pick para reword e salve o arquivo
3. O git automaticamente abrirá os reword para edição
4. Salve e seja feliz enviando as alterações para o repositório:  
`git push origin master -f`

# E se eu quiseser baixar um projeto do github ?

- 
- `git clone [url]`
- 

O comando `git clone` faz uma cópia do projeto em nossa máquina.

# O que é um readme.md?

- Nada mais é do que um arquivo de texto onde você poderá descrever, documentar e exemplificar, o seu projeto, algumas informações presentes nele são:

1. Título e breve descrição do projeto;
2. Pré-requisitos;
3. Guia de instalação;
4. Informação de copyright e licenciamento;
5. Autoria e contribuições;





**Ele é um arquivo obrigatório? Não! Se não tiver um arquivo README.md meu projeto continuará a funcionar? Sim!**



MINICURSO



# Por que escrever um bom readme.md?





## Criando um **readme.md**

1. *Digite no bash: touch readme.md*

# Algumas boas práticas no uso do git

- Escreva commits pequenos e informativos
- Escreva um bom readme.md
- Organize as pastas de seu projeto

# Usando o git através de interfaces visuais



# O MERCADO procura quem SABE GIT

DESEJÁVEL CONHECIMENTO EM:

Sass ou Less

React

Grunt, Gulp e WebPack

Familiaridade com sistemas de controle de versão (SVN e Git)

REQUISITOS:

- Java 8

- Spring Boot

- Maven

- API Restful

- Tomcat

- Docker

- Sql Server

- GIT

- JUnit / Mockito

Desejável:

Conhecer JAVA JBoss

Infra AWS

Experiência com ferramentas de versionamento de código (GIT,...);

Ser auto-didata;

# Para saber mais...

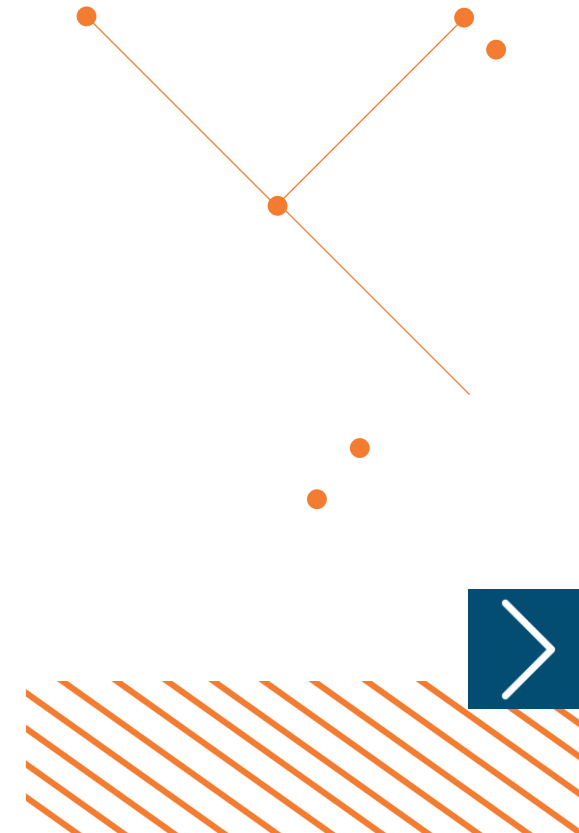
- [Git Avançado](#)
- [Git How To](#)
- [GUI GitKraken](#)
- [GUI SourceTree](#)
- [Guia Prático](#)
- [Configurando chave SSH](#)



0 0 0  
0 1 0  
1 1 0  
1 0  
0  
1  
0

# RESUMINHO DO QUE VIMOS

- O que é um readme.md
- Porque escrever um bom readme.md
- Boas práticas no uso do git
- Usando o git através de interfaces visuais







**OBRIGADA!**



000  
010  
110  
10  
0  
1  
0

MINICURSO



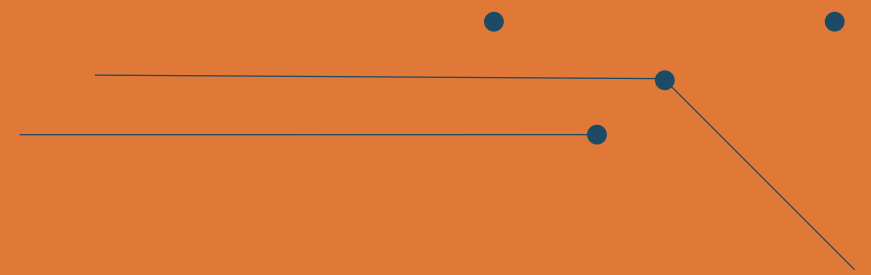
# Contatos:

**NÁDIA OLIVEIRA**

E-mail - [nadiaaoliver@gmail.com](mailto:nadiaaoliver@gmail.com)

LinkedIn - [nadiaaoliver](#)

Github - [NadiaaOliver](#)



**2019**  
SACSI