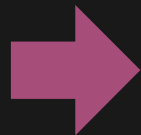




# Desarrollo colaborativo con Git & Github

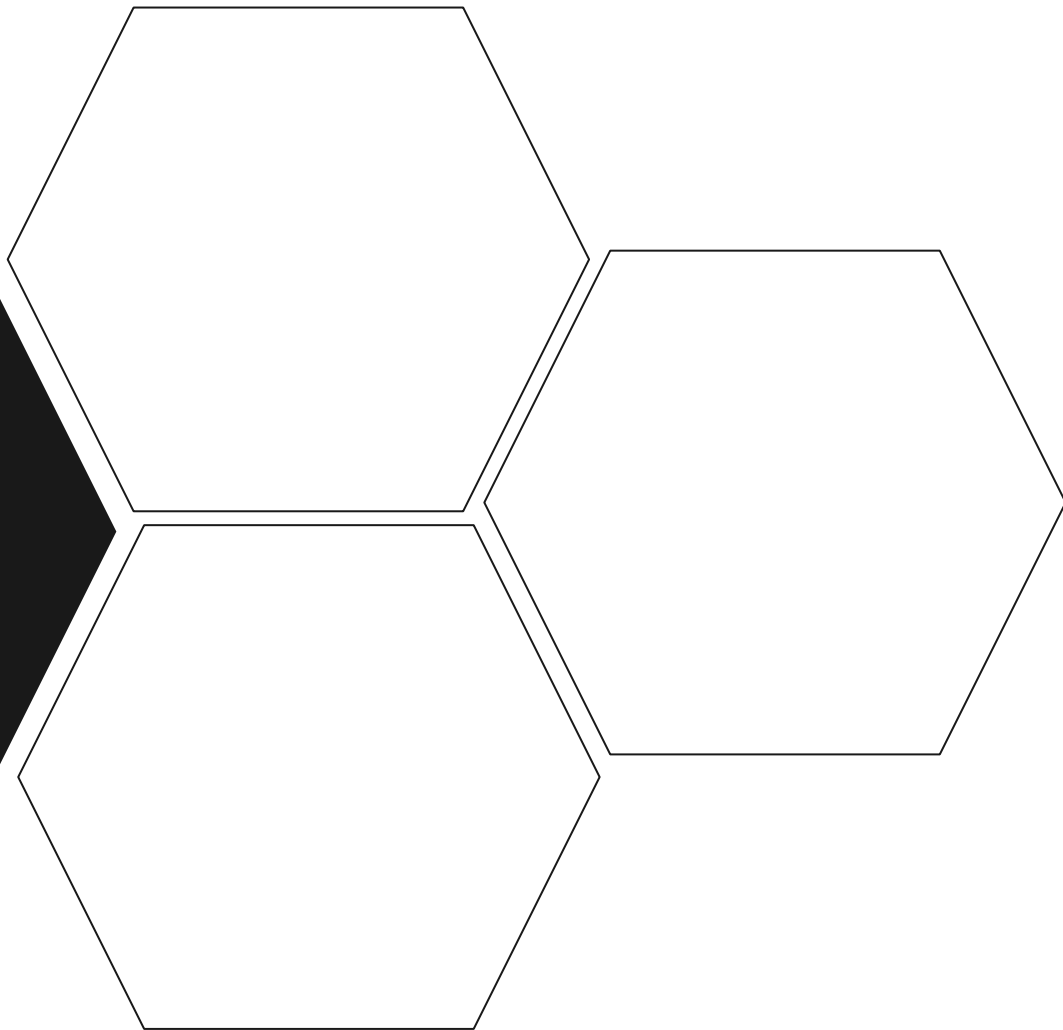
CLASE 1

## Distribución del contenido del curso



### CLASE 1

Aprenderemos cómo  
avanzar en la línea  
de tiempo. También  
como subir y bajar  
cambios en la nube.

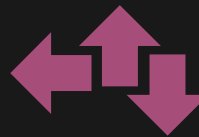


## Distribución del contenido del curso



### CLASE 1

Aprenderemos cómo avanzar en la línea de tiempo. También como subir y bajar cambios en la nube.



### CLASE 2

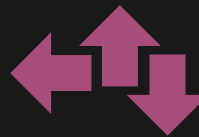
Aprenderemos cómo retroceder en la línea de tiempo e ir hacia los costados en líneas de tiempo paralelas.

## Distribución del contenido del curso



### CLASE 1

Aprenderemos cómo avanzar en la línea de tiempo. También como subir y bajar cambios en la nube.



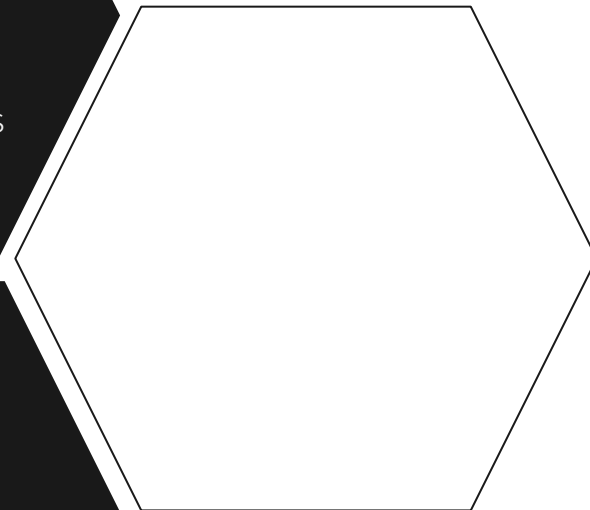
### CLASE 2

Aprenderemos cómo retroceder en la línea de tiempo e ir hacia los costados en líneas de tiempo paralelas.

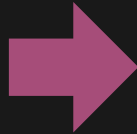


### CLASE 3

Aprenderemos cómo retroceder en la línea de tiempo e ir hacia los costados en líneas de tiempo paralelas.

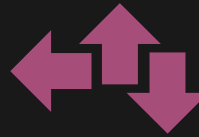


## Distribución del contenido del curso



### CLASE 1

Aprenderemos cómo avanzar en la línea de tiempo. También como subir y bajar cambios en la nube.



### CLASE 2

Aprenderemos cómo retroceder en la línea de tiempo e ir hacia los costados en líneas de tiempo paralelas.



### CLASE 3

Aprenderemos cómo retroceder en la línea de tiempo e ir hacia los costados en líneas de tiempo paralelas.

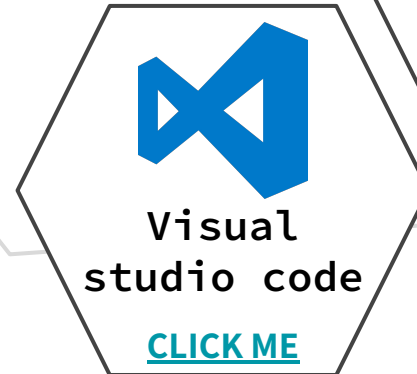


### CLASE 4

Aprenderemos funcionalidades complementarias en general de Git & Github.

## EJERCICIO 0

Instalar las siguientes tecnologías para poder hacer el curso. Mientras seguimos con la teoría.



# OBJETIVOS DE LA CLASE 1

- ❑ ¿Qué son los sistemas de control de versionado?
- ❑ ¿Qué es un repositorio?
- ❑ ¿Qué es Git y GitHub?
- ❑ ¿Cuáles son los estados de Git?
- ❑ Comandos básico



## COMANDOS DE LA CLASE DE HOY

Los alumnos pueden utilizar esta lista para guiarse en qué comando se explicó en cada clase, para saber si corresponde ver este video grabado o no. Si no llega a estar en esta clase, seguro en la siguiente.

- ❏ `git clone`
- ❏ `git init`
- ❏ `git remote`
- ❏ `git config --global user.name <userName>`
- ❏ `git config --global user.email <email>`
- ❏ `git status`
- ❏ `git add .`
- ❏ `git add -u`
- ❏ `git commit -m "mensaje"`
- ❏ `git commit`
- ❏ `git push`
- ❏ `git pull`



# DOCUMENTACIÓN PARA ESTE CURSO



Alumni

[CLICK ME](#)



Github

[CLICK ME](#)



Git

[CLICK ME](#)


Nota: Utilizar la versión en inglés para Git porque la traducción al español tiene varios errores de concepto.

Nota: Alumni → Mis cursos → Git : Desarrollo colaborativo → Acceso al contenido

# 01

## INTRODUCCION





**¿Cual es el  
problema que  
buscaba resolver Git  
& Github?**



**TP-Kernel-V1.2.3**



**TP-Kernel-V2.2-FINAL**



**TP-Kernel-V2.2-FINAL FINAL**



**TP-Kernel-V2.2-FINAL POSTA**



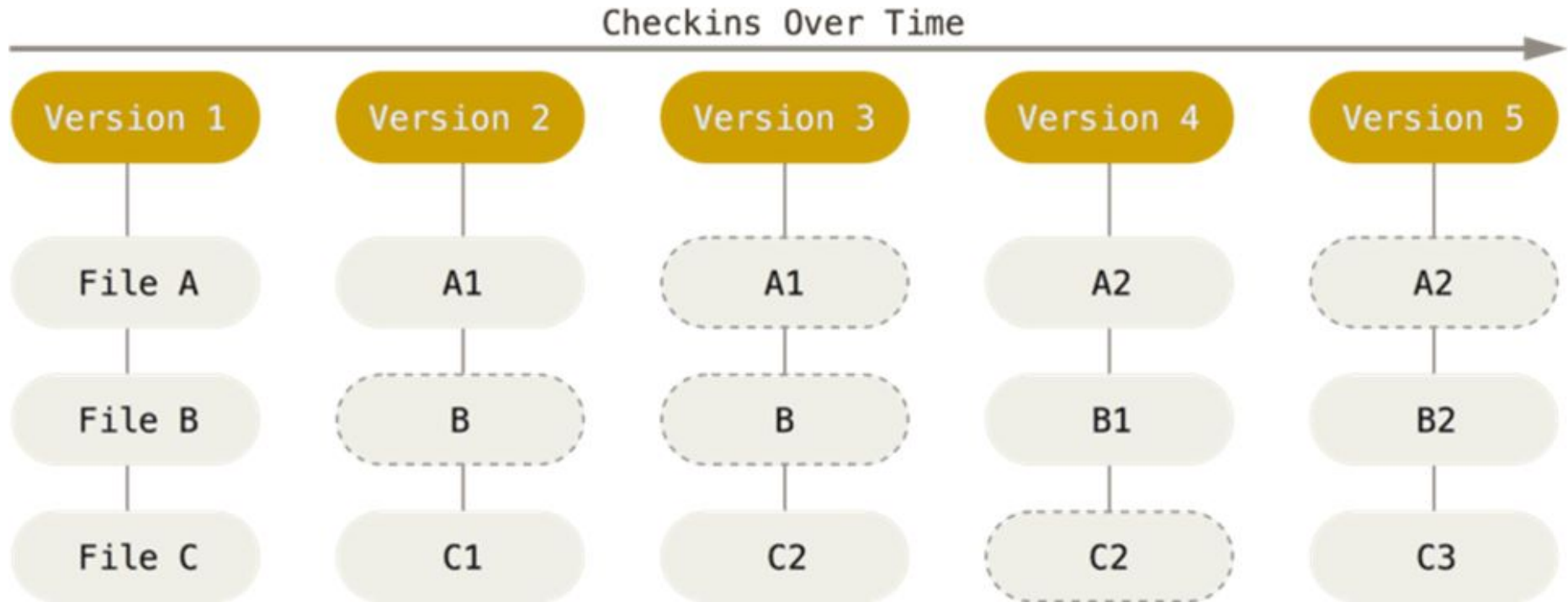
**TP-Kernel-V2.2-FINAL POSTA DE VERDAD**

¿Cuál es la versión  
funcional a entregar?

Un SISTEMA DE CONTROL DE VERSIONADO es un software que realiza el seguimiento de los cambios de un conjunto de archivos.

Básicamente lo que hace es construir una línea de tiempo que permite recuperar estas versiones anteriores de estos archivos en cualquier momento.

Un ejemplo de esa línea de tiempo.

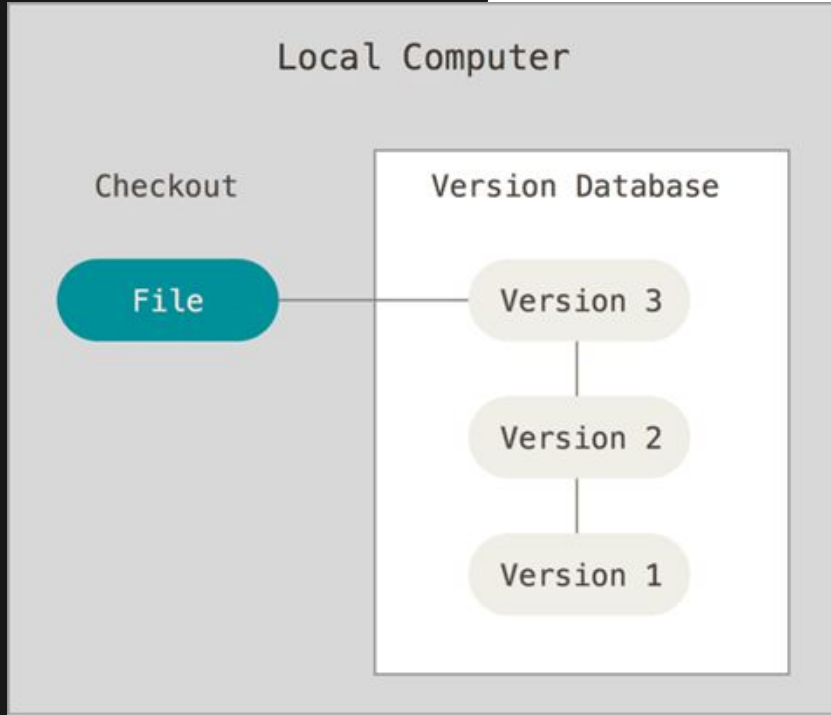


# 02

## TIPOS DE SVC



# LOCALES



## VENTAJAS

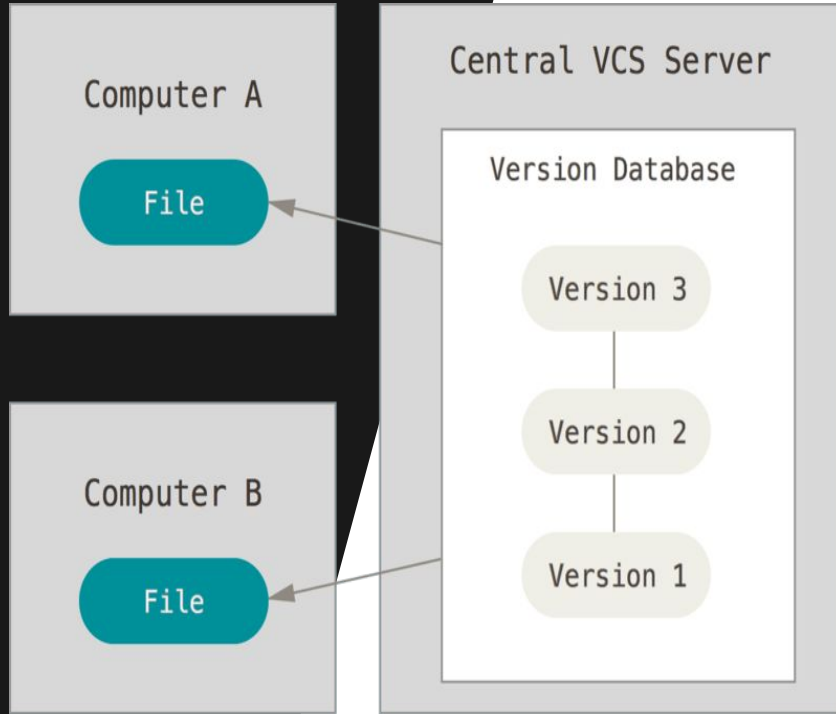
- ❑ Disponibilidad del 100% porque no requiere internet.
- ❑ Es muy sencillo de utilizar.

## DESVENTAJAS

- ❑ Sin tolerancias a fallos, no existe backup.
- ❑ No permite el trabajo colaborativo.



# CENTRALIZADOS



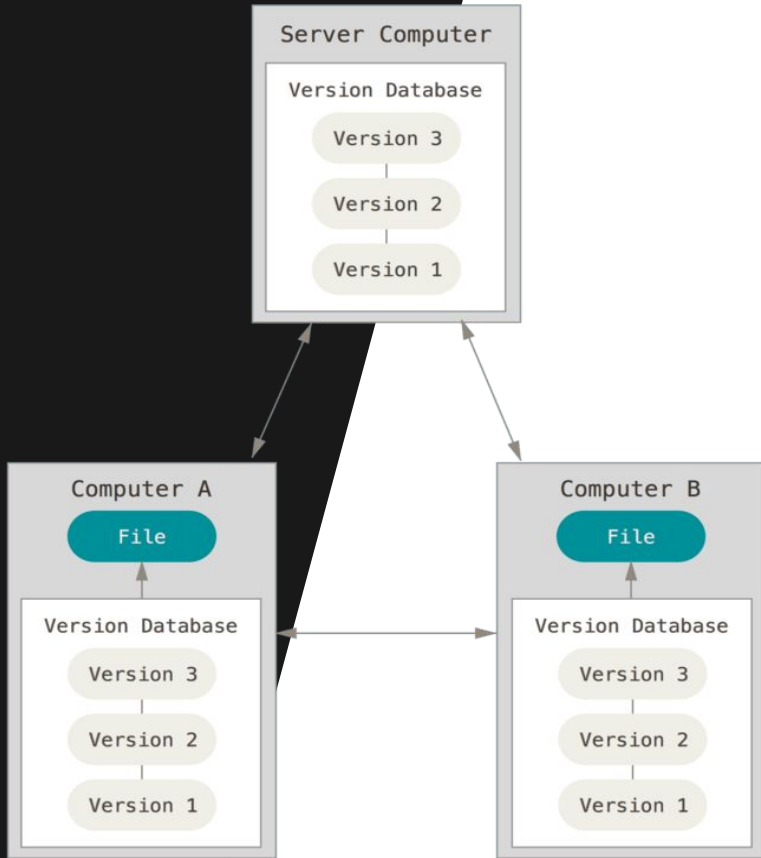
## VENTAJAS

- ❑ Permite el trabajo colaborativo.
- ❑ Es muy sencillo de utilizar.

## DESVENTAJAS

- ❑ Sin tolerancias a fallos, no existe backup.
- ❑ Depende de internet para poder trabajar.

# DISTRIBUIDO



## VENTAJAS

- ❑ Permite el trabajo colaborativo.
- ❑ Disponibilidad del 100%. Se trabaja localmente y luego se actualiza la versión en el servidor cuando tengamos internet.
- ❑ Tiene tolerancia a los fallos, porque existen múltiples copias.

## DESVENTAJAS

- ❑ No es tan sencillo de utilizar porque existen múltiples copias de un mismo proyecto.

# 03

## GIT & GITHUB





**Git, ¿De qué tipo de  
SVC es?**



# git

Es un sistema de control de  
versionado **DISTRIBUIDO**.

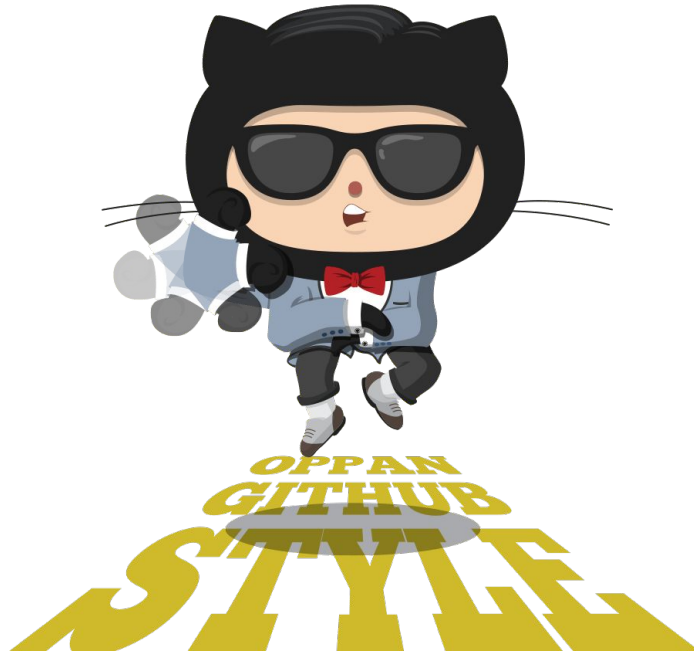
Github es un servidor que  
utiliza Git, y se utiliza como la  
versión centralizada de git.

# GitHub



**¿Qué otros usos  
tiene Github?**

Github tiene un par de funcionalidades más.




Es como una red  
social para  
programadores,  
pero en vez de  
publicar lo que  
comes ...





```
.function(b, c) {  
    var d = a(c.form.querySelectorAll("input[type=checkbox]")[name] + ".val" + "T");  
    if (0 == d.index(0.el)) {  
        var e = d.filter(":checked").length;  
        return e >= b.arg || g.minChecked.replace("{count}", b.arg)  
    }  
},  
  
maxSelected: function(a) {  
    return null != a.val ? a.val.length <= a.arg || g.maxSelected.replace("{count}", a.arg) : null  
},  
  
minSelected: function(a) {  
    return null != a.val && a.val.length >= a.arg || g.minSelected.replace("{count}", a.arg)  
},  
  
radio: function(b) {  
    var c = a(this.form.querySelectorAll("input[type=radio]")[name] + ".val" + "T").filter(":checked").length;  
    return 1 == c  
},  
  
custom: function(a, b) {  
    var c = b.options.custom[a.arg],  
        d = new RegExp(c.pattern);  
    return d.test(a.val) || c.errorMessage  
},  
  
remote: function(a) {  
    a.remote = a.arg  
}  
};  
  
b = function(b, c) {  
    this.handler = 1,  
    this.prototype = {  
        b: b,  
        c: c,  
        extend: function(f, i, h, c) { this.foo = b, this.do = c, this.bar() },  
        foo: function() { this.foo = b, this.do = c, this.bar() },  
        bar: function() { this.foo = b, this.do = c, this.bar() }  
    };  
    this.extend(0, 1, h, c);  
    this.foo();  
    this.bar();  
}
```



**¿Existen otras  
alternativas a  
Github?**

## ALTERNATIVAS A GITHUB, NO A GIT



Gitlab

[CLICK ME](#)



Bitbucket

[CLICK ME](#)

## ALTERNATIVA A GIT



[TortoiseSVN](#)

TortoiseSVN

[CLICK ME](#)

## EJERCICIO 1

Crear un cuenta en Github. La necesitamos para poder configurar Git localmente, por lo que necesitarás el email, nombre, y la contraseña.

Notas:


- ❑ El nombre de usuario se puede cambiar después de crear la cuenta.
- ❑ A una cuenta de Github se le puede asociar varios emails simultáneamente.
- ❑ Se puede desasociar un email y utilizarlo para abrir una nueva cuenta totalmente independiente.
- ❑ Si tiene un email con extensión educativa (edu) pueden habilitar por un año una cuenta git pro.





**¿Que es un  
repositorio?**

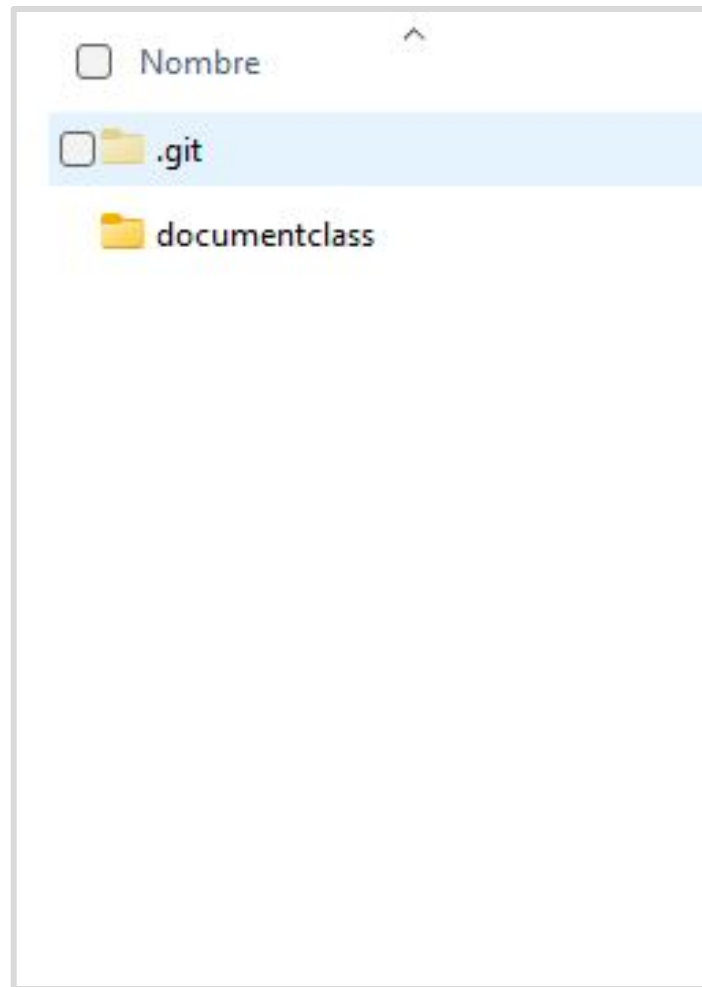
Un REPOSITORIO es una carpeta donde se realiza el seguimiento o no de los cambios realizados sobre un conjunto de archivos. Existe una copia en cada computadora de un colaborador y otro en la nube.




**¿Cual es la  
diferencia entre  
repositorio y una  
carpeta cualquiera?**

## No toques esa carpeta

Existe una carpeta oculta .git es donde vive la línea de tiempo de nuestro proyecto.







**¿Qué tipos de  
repositorios  
existen?**

# Gratis y Publicos

## VENTAJAS

- ❑ No tiene ningún tipo de costo.
- ❑ Acceso al 100% de las funciones.

## DESVENTAJAS

- ❑ El código está a la vista de toda la comunidad, registrada o no en Github.



## Gratuitos y Publicos

### VENTAJAS

- ❑ No tiene ningún tipo de costo.
- ❑ Acceso al 100% de las funciones.

### DESVENTAJAS

- ❑ El código está a la vista de toda la comunidad, registrada o no en Github.

## Gratuitos y Privados

### VENTAJAS

- ❑ No tiene ningún tipo de costo.
- ❑ El código solo es visible para los colaboradores

### DESVENTAJAS

- ❑ Funciones limitadas.
- ❑ Máximo 3 colaboradores.



## Gratuitos y Públicos

### VENTAJAS

- ❑ No tiene ningún tipo de costo.
- ❑ Acceso al 100% de las funciones.

### DESVENTAJAS

- ❑ El código está a la vista de toda la comunidad, registrada o no en Github.

## Gratuitos y Privados

### VENTAJAS

- ❑ No tiene ningún tipo de costo.
- ❑ El código solo es visible para los colaboradores

### DESVENTAJAS

- ❑ Funciones limitadas.
- ❑ Máximo 3 colaboradores.

## Privados y Pagos

### VENTAJAS

- ❑ El código solo es visible para los colaboradores..
- ❑ Acceso al 100% de las funciones.

### DESVENTAJAS

- ❑ El costo depende de la cantidad de colaboradores.

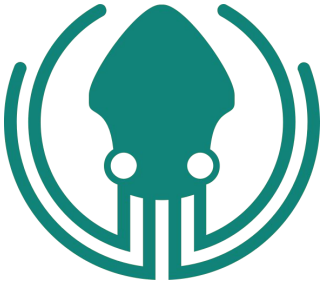




# Consola vs GUI

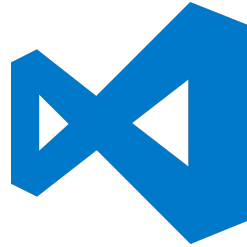
## GUI

Las empresas aman las GUI porque la curva de aprendizaje y la cantidad de errores son menores.



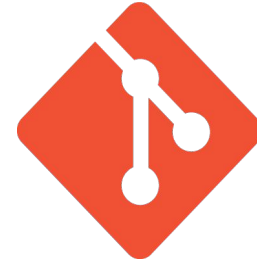
**Gitkraken**

Multiplataforma



**Visual studio  
code**

Multiplataforma



**Git Desktop**

Multiplataforma



**Source tree**

Windows y Mac



**¿Cuáles son los  
niveles de  
configuración?**

## **.gitconfig**

Git tiene varios archivos *.gitconfig* que indica cómo debe funcionar el programa.

### **LOCAL**

Solo afecta un repositorio, y existe uno en cada repositorio.

### **GLOBAL**

Afecta a todos los repositorios del usuario, y existe uno solo por usuario de la computadora.

### **SYSTEM**

NO TOCAR. Existe uno por usuario de computadora y contiene información delicada.



## EJERCICIO 2

Configuramos nuestro entorno de trabajo antes de hacer cualquier otra cosa.



```
1 git config --nivel variable <valor>
2 #Configuramos
3 git config --global user.name <user_name>
4 git config --global user.email <user_email>
5 #Confirmamos
6 git config --global -l
7 #Borramos si nos equivocamos
8 git config --global --unset <variable_nombre>
```



**¿Cómo se crea un  
repositorio?**

## EXISTEN DOS CAMINOS

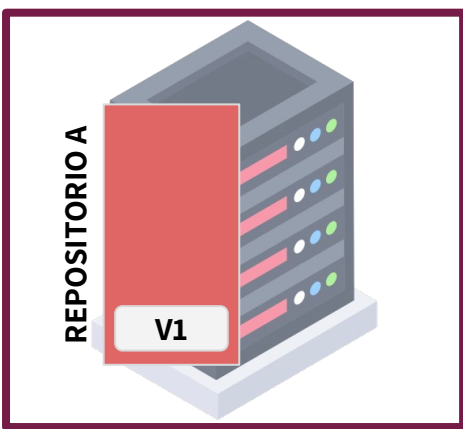


## EJERCICIO 3

Creamos un repositorio desde Github, y luego lo vamos a clonar en nuestra computadora.



Github (En la nube)



## PASOS PARA CLONAR

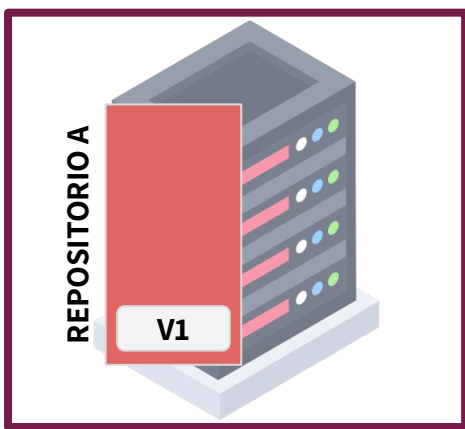
### PRIMER PASO - En Github

Ir a Github y crear un nuevo repositorio desde la interfaz gráfica.

Tu compu (Local)



Github (En la nube)



## PASOS PARA CLONAR

### PRIMER PASO - En Github

Ir a Github y crear un nuevo repositorio desde la interfaz gráfica.

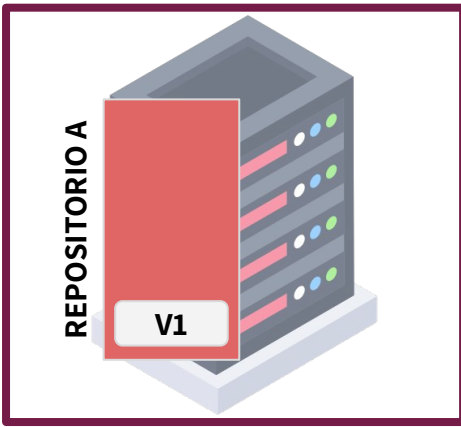
### SEGUNDO PASO - En Github

Copiar la url de clonación que nos crea Github automáticamente.

Tu compu (Local)



Github (En la nube)



Tu compu (Local)



## PASOS PARA CLONAR

### PRIMER PASO - En Github

Ir a Github y crear un nuevo repositorio desde la interfaz gráfica.

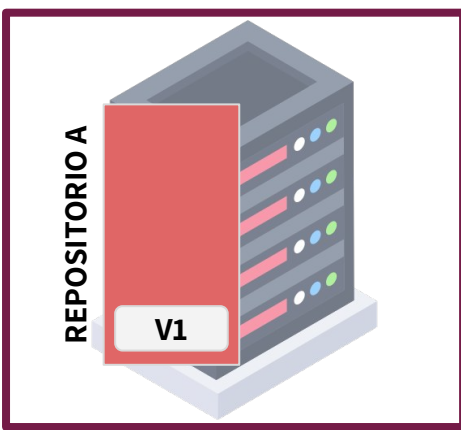
### SEGUNDO PASO - En Github

Copiar la url de clonación que nos crea Github automáticamente.

### TERCER PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

Github (En la nube)



Tu compu (Local)



## PASOS PARA CLONAR

### PRIMER PASO - En Github

Ir a Github y crear un nuevo repositorio desde la interfaz gráfica.

### SEGUNDO PASO - En Github

Copiar la url de clonación que nos crea Github automáticamente.

### TERCER PASO - En Consola

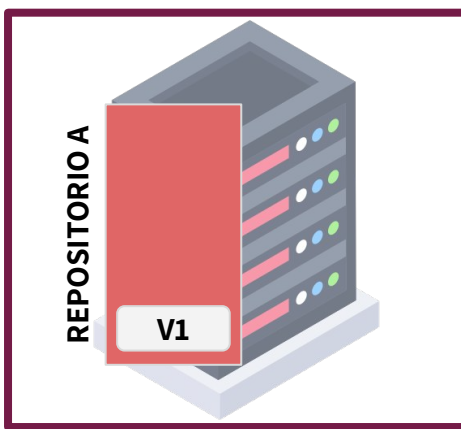
Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### CUARTO PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.



Github (En la nube)



Tu compu (Local)



## PASOS PARA CLONAR

### PRIMER PASO - En Github

Ir a Github y crear un nuevo repositorio desde la interfaz gráfica.

### SEGUNDO PASO - En Github

Copiar la url de clonación que nos crea Github automáticamente.

### TERCER PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### CUARTO PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### QUINTO PASO - En Consola

Ejecutamos el siguiente comando con la url copiada de Github.

```
git clone <url>
```

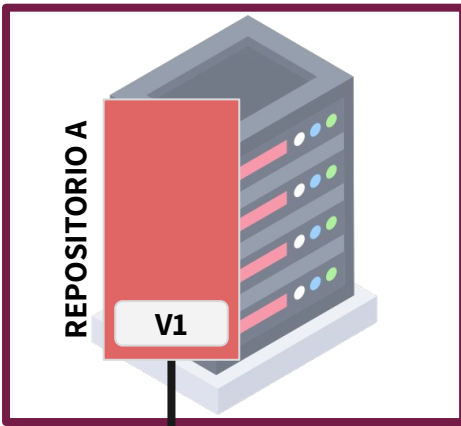
LOCAL EN TU COMPUTADORA

GITHUB

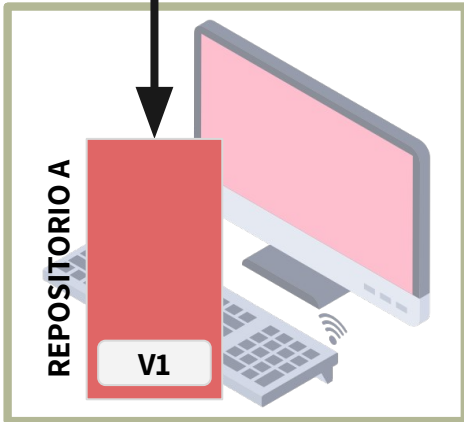
`git clone <url>`

Solo lo hacemos la primera vez  
para obtener nuestra copia de la  
línea de tiempo.

Github (En la nube)



Tu compu (Local)



## PASOS PARA CLONAR

### PRIMER PASO - En Github

Ir a Github y crear un nuevo repositorio desde la interfaz gráfica.

### SEGUNDO PASO - En Github

Copiar la url de clonación que nos crea Github automáticamente.

### TERCER PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### CUARTO PASO - En Consola

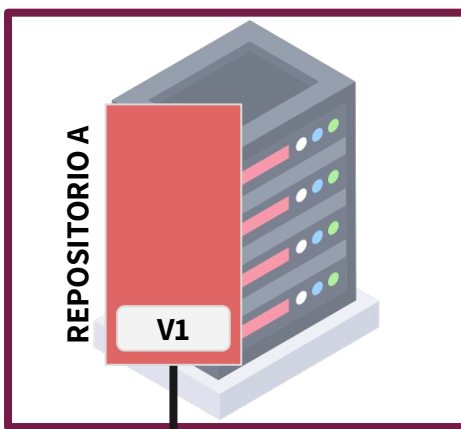
Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### QUINTO PASO - En Consola

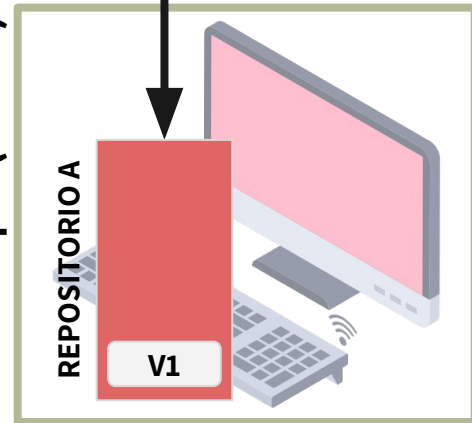
Ejecutamos el siguiente comando con la url copiada de Github.

```
git clone <url>
```

Github (En la nube)



Tu compu (Local)



## PASOS PARA CLONAR

### PRIMER PASO - En Github

Ir a Github y crear un nuevo repositorio desde la interfaz gráfica.

### SEGUNDO PASO - En Github

Copiar la url de clonación que nos crea Github automáticamente.

### TERCER PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### CUARTO PASO - En Consola

Abrir una consola en la ubicación donde deseamos clonar el repositorio. Podemos abrir una consola con Click derecho y la opción “*git bash here*”.

### QUINTO PASO - En Consola

Ejecutamos el siguiente comando con la url copiada de Github.

```
git clone <url>
```

### SEXTO PASO - En Consola

Abrimos el repositorio en Visual studio code para comenzar a trabajar.

```
code <nombre_repositorio>
```



# Estados de git

**LOCAL EN TU COMPUTADORA**

**GITHUB**

## LOCAL EN TU COMPUTADORA

WORKING

UNTRAKING

TRACKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

Son todos los archivos con los que estamos trabajando. Son los cambios que más adelante convertiremos en una nueva versión de mi línea de tiempo.

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

Son todos los archivos nuevos de los cuales Git todavía no hace un seguimiento de sus cambios.



## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

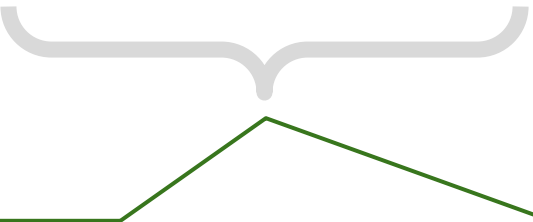
UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB



Son todos los archivos que ya están en seguimiento, y por lo tanto Git está atento a los cambios que realicemos.

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

Son los archivos que NO sufrieron ningún cambio respecto a la última versión (commit) creada.

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

Son los archivos que SÍ sufrieron algún cambio respecto a la última versión (commit) creada.

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

Son los elegidos previamente de WORKIN para que formen parte de esa nueva versión que estamos por crear.

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

Cuando pasamos los archivos de STAGING a REPOSITORY, creamos una nueva versión en nuestra línea de tiempo, con un número de identificación único, y acompañado de un comentario que detalla qué cambios se introdujeron.

**Un COMMIT es una foto del estado del proyecto en un momento determinado a la cual podemos volver si es necesario. Cuando decido crear un commit, estamos creando una nueva versión en la línea de detalle.**

**Cuando hablamos de ir a un commit, estamos deseando volver a un estado del proyecto guardado previamente.**

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

Todo lo que ocurra de este lado, se puede borrar sin dejar rastros.

Lo subido a la nube, no se puede borrar, solo se puede compensar.

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED


STAGING

REPOSITORY

GITHUB

Cualquier conflicto se  
resuelve a este nivel. No  
en la nube.





**¿Cómo es un flujo de  
trabajo ideal? Sin  
errores por ahora.**

# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git clone <url>`



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git status`

`git clone <url>`



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git status`

`git clone <url>`

`git add .`



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

git status

git clone <url>

git add .

git status



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

git status

git clone <url>

git add .

git status

git commit -m "mensaje"

# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

git status

git clone <url>

git add .

git status

git commit -m "mensaje"

git log



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

git status

git clone <url>

git add .

git status

git commit -m "mensaje"

git log

git push



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git status`

`git clone <url>`

`git add .`

`git status`

`git commit -m "mensaje"`

`git log`

`git push`



# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git status`

`git pull`

`git add .`

`git status`

`git commit -m "mensaje"`

`git log`

`git push`





**Flujos alternativos e  
inconvenientes.**

# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

```
git add <file_name>
```

```
git add -u <file_name>
```

```
git add -u .
```

# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

```
git add <file_name>
```

```
git add -u <file_name>
```

```
git add -u .
```

```
git add <expresion_regular>
```

# LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git commit -m "mensaje"`

`git commit`



Editor texto  
de consola

[CLICK ME](#)

# OPERACIONES BÁSICAS CON VIM



Editor texto  
de consola

[CLICK ME](#)

## Habilitar modo de escritura

Solo tienes que presionar  
la tecla “w”.

## Guardar y salir del editor

Primero presionas “ESC”,  
luego escribes “:wq” y por  
último “ENTER”.

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git log`

Muestra todos los commit con detalle completo en la rama.

`git log -2`

Muestra solo los 2 primeros commit con detalle completo en la rama.

`git log -<N>`

Muestra solo los N primeros commit con detalle completo en la rama.

`git log --oneline`

Muestra todos los commit con un detalle reducido en la rama.



## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

`git log`

Muestra todos los commit con detalle completo en la rama.

`git log -2`

Muestra solo los 2 primeros commit con detalle completo en la rama.

`git log -<N>`

Muestra solo los N primeros commit con detalle completo en la rama.

`git log --oneline`

Muestra todos los commit con un detalle reducido en la rama.

## LOCAL EN TU COMPUTADORA

WORKING

TRACKING

UNTRAKING

UNMODIFIED

MODIFIED

STAGING

REPOSITORY

GITHUB

```
git log
```

```
git log -2
```

```
git log -<N>
```

```
git log --oneline
```

```
git log --oneline -<N>
```

¿Este comando que hace?



FIN DE LA  
TEORÍA

[rapa.matias.e@gmail.com](mailto:rapa.matias.e@gmail.com)  
Asunto : CURSO/DIA/TURNO