Introduccion a Sistema de control de versiones.

Un sistema de control de versiones es una herramienta que permite gestionar los cambios en el código de un proyecto a lo largo del tiempo. Es independiente del lenguaje de programación. Git es un sistema de control de versiones distribuido, y GitHub es una plataforma en línea que facilita la colaboración y el alojamiento de repositorios de Git.

Git

Git permite a los desarrolladores registrar y administrar cada modificación en su código. Algunas características de Git son:

- **Seguimiento de cambios**: Git guarda cada cambio en el código, lo que permite volver a versiones anteriores si es necesario.
- **Distribución**: A diferencia de otros sistemas, cada copia de un repositorio Git es completa y contiene toda la historia del proyecto. Esto hace que Git sea rápido y robusto.
- **Ramas y fusiones**: Git permite trabajar en ramas (branches), lo que facilita el desarrollo paralelo. Los cambios pueden fusionarse en la rama principal cuando están listos.
- **Colaboración**: Git facilita la colaboración al permitir a los desarrolladores trabajar en sus propias copias del código y luego combinar los cambios.

GitHub

GitHub es una plataforma en la nube para alojar repositorios Git y colaborar en ellos. Ofrece herramientas adicionales que complementan a Git:

- **Interfaz en línea**: GitHub permite ver, editar y comentar el código directamente desde el navegador.
- **Gestión de proyectos**: GitHub ofrece herramientas para administrar tareas, como el seguimiento de problemas (issues), solicitudes de cambio (pull requests), y la revisión de código.
- **Colaboración social**: Los repositorios pueden ser públicos o privados, y GitHub facilita la colaboración entre equipos al permitir comentarios y sugerencias en el código.
- **Automatización y CI/CD**: GitHub Actions permite configurar flujos de trabajo automáticos, como pruebas automáticas o despliegues.

En resumen: Git se enfoca en gestionar versiones de código de manera distribuida, mientras que GitHub facilita el almacenamiento en la nube, la colaboración, y el manejo de proyectos sobre la base de un repositorio Git.

Manos a la obra

Algunos de los pasos que debemos realizar, se deben a la version antigua de Visual Studio que tienen los equipos del talles, la version 2012 no tiene integrado el control de versiones, con lo cual se debe recurrir a herramientas externas. Como Git Bashm cuyo comandos son los mismos para Windows, Linux y Mac.

1. Instalacion Configuracion inicial

Estos pasos solo se realizan la primera vez.
Como las computadoras del tallller están "frizadas", los pasos que requieran cambios en el equipo local deben realizarse nuevamente en cada inicio de clase.

1.1 Crear Cuenta GitHub

En el sitio https://github.com/
Crar una cuenta con un correo electrónico válido.

1.2 Descargar Git

Del sitio https://git-scm.com/

Descargar version del sistema operativo

Una vez instalado Git. Se dispone de 3 opciones, desde el menú inicio: Git Bash

Esto es una ventana de comando en modo texto, no es la mas amigable, pero se tiene la mayor cantidad de opciones para utilizar.

1.3 Configurar cuenta Git local

Al instalar Git por primera vez y usarlo desde la línea de comando, hay algunas configuraciones básicas que te ayudarán a identificarte en cada repositorio y a personalizar algunos aspectos del uso de Git. Aquí te detallo los pasos:

1. Configurar el nombre de usuario y correo electrónico

Esto es importante porque cada commit que hagas en Git estará asociado a estos datos. Los configuraremos de la siguiente manera:

```
git config --global user.name "Tu Nombre"
git config --global user.email "tuemail@example.com"
```

```
Nota: La opción --global hace que esta configuración se aplique a todos los
repositorios de tu sistema. Si quieres que solo se aplique a un proyecto
específico, omite --global.
```

2. Verificar la configuración

Para asegurarte de que las configuraciones se guardaron correctamente, puedes ejecutar:

```
git config --list
```

Uso habitual en un SCM

Crear un repositorio en GitHub

En un sistema de control de versiones (SCM) como GitHub, un repositorio es un espacio o contenedor donde se almacena el código de un proyecto y su historial de cambios. Básicamente, es el lugar donde se guarda y organiza toda la información y el desarrollo de un proyecto, permitiendo que los usuarios puedan colaborar, revisar y mejorar el código en equipo.

En la pagina de github, para crear un repositorio solo debe presional el boton `New`.
En el campo repository name debe colocal el nombre del repositorio. Aunque no es obligatorio, es recomendable que se utilice el mismo nombre que el proyecto, en este caso, el de Visual Studio.

En los ejemplos el proyecto es llamado proyecto1. En esta actividas los alumnos deben colocar como nombre: Grupo-. Por ejemplo Grupo1-Izaguirre. Con la finalidad de poder identificar al grupo y al menos uno de los integrantes.

Crear proyecto en Visual Studio

En Visual Studio crar un proyecto, esté es el que se subirá al control de versiones, para poder recuperar en las proximas clases.
Tener en cuena que el nombre que se elija, será el nombre del la carpeta contenedora y el nombre recomendado para el repositorio asociado.

En el explorador de archivos dirigirse a la ruta del proyecto, para obtener la ruta completa del proyecto.
Abrir una consola git Bash y tambien dirigirse a la ruta base del proyecto. Normalmente donde se encuentra el archivo .sln.

```
cd "C:\Users\joriz\source\repos\proyecto1"
```

```
Nota: las comillas solo son necesarias si alguno de los nombres contiene espacios.
```

Comando para crear un prepositorio local.

Estando posicionado en la ruta base del proyecto escribir

```
git init
```

Consultar estado del repositorio

```
git status
```

Muestra detalle de los archivos del repositorio. Sin seguimiento, modificados y nuevos.

Selecctionar los archivos a incluir.

```
git add .
```

```
El punto indica todos, pero puede seleccionarse el archivo que se necesite,
tambien se pueden usar comodines como * y ? .


Si indica acceso denegado a almenos 1 archivo, eso depende de la version del
Visual Studio y suele suceder la primera vez que se seleccionan archvios, para
poder continuar se debe cerrar visual Studio y volver a ejecutar el comando. Luego
de ejecutado el comando se debe ingresar nuevamenente al proyecto en Visual
Studio.
```

Confirmar los cambios

El comando git commit en Git se utiliza para guardar los cambios que has realizado en tu proyecto, registrándolos en el historial del repositorio. Al hacer un commit, se crea un "punto de control" o "instantánea" del estado actual del proyecto, lo cual permite volver a esta versión en el futuro si es necesario.

```
git commit -m"Primer commit del proyecto"
```

Subir los cambios por primera vez

Estos tres comandos de Git se utilizan en la configuración inicial de un proyecto para establecer la rama principal, asociar el repositorio local con un repositorio remoto, y subir el proyecto a la plataforma de control de versiones (por ejemplo, GitHub). Aquí tienes una explicación detallada de cada uno:

```
git branch -M main
git remote add origin https://github.com/joriza/proyecto1.git
git push -u origin main
```

1. `git branch -M main`

Este comando:

- **Renombra la rama actual** a `main`. Si el nombre de la rama actual es `master` (el nombre por defecto en versiones más antiguas de Git), este comando la cambia a `main`, que es la convención moderna, especialmente en plataformas como GitHub.
- `-M` es una opción que **fuerza el cambio de nombre**, incluso si ya existe una rama llamada `main`. La `M` de la opción significa "move" o "renombrar" de forma forzada.

2. `git remote add origin https://github.com/joriza/proyecto1.git`

Este comando:

- **Conecta el repositorio local con un repositorio remoto** ubicado en la URL especificada (en este caso, `https://github.com/joriza/proyecto1.git`).
- `origin` es el alias que Git usa por convención para referirse al repositorio remoto principal, aunque puedes darle otro nombre si lo prefieres.
- Al agregar este "remoto", permites que Git sincronice los cambios entre tu repositorio local y el repositorio en GitHub, usando `origin` como nombre de referencia para la URL del remoto.

3. `git push -u origin main`

Este comando:

- **Sube la rama** `main` del repositorio local al repositorio remoto (`origin`), compartiendo todos los commits y el historial de cambios que tenga hasta el momento.
- La opción `-u` (o `--set-upstream`) establece la rama `origin/main` (la rama `main` en el repositorio remoto) como la "upstream" o rama de seguimiento para la rama `main` local. Esto significa que en el futuro, al usar `git push` o `git pull`, Git sabrá a qué rama remota referirse sin necesidad de especificarla.

Después de ejecutar estos comandos, tu proyecto estará disponible en GitHub (o el servicio que estés usando), y podrás seguir colaborando y sincronizando cambios entre el repositorio local y el remoto sin tener que redefinir la rama ni la URL.

Registrar modificaciones del código

Habitualmente, en una terminal, estando en la ruta base del proyecto, con estos comandos es suficiente en el 99% de los casos.

```
git add .
git commit -m"Mensaje del commit"
git push
```

Restaurar un proyecto desde GitHub

En la carpeta repos del Visual Studio

```
git clone https://github.com/joriza/proyecto1.git
```

Este comando descarga de GitHub el repositorio completo y se podrá abrir el proyecto desde Visual Studio.