Se muestra cómo usar Visual Studio 2022 junto con GitHub para hacer trabajo colaborativo y control de versiones.

Se enfatiza que para esto es necesario tener una cuenta de GitHub previamente.

También se explica que Visual Studio debe estar configurado para reconocer Git y trabajar con repositorios remotos.

Se debe de crear o tener una cuenta activa en GitHub, el tipo de autenticación va a depender de donde se acceda.

En Visual Studio se debe de configurar con la cuenta de GitHub para poder usar el control de versiones y verificar que las opciones y paquetes necesarios se encuentren instalaciones y seleccionados.

Posteriormente se debe de crear un Repositorio desde Visual Studio como Git Local, luego se conecta con un repositorio remoto en GitHub .

Si no se quiere hacer así, se puede clonar un repositorio remoto y traerlo a la maquina local a través de Visual Studio.

Luego de hacer cambios en el código local, se hace un commit para guardar los cambios de forma local, se debe de agregar un mensaje en el commit de forma clara para que no haya ambigüedad.

Luego de esto se realiza un Push para subir los cambios al repositorio remoto.

Para traer cambios que hayan realizado otros, se realiza un Pull / Fetch / Merge.

Para incorporar los cambios que otros hicieron, se puede usar pull o fetch + merge.

Fetch trae los cambios del servidor remoto; merge los integra con los cambios locales.

Si hay conflictos, Visual Studio ofrece herramientas visuales para compararlos y resolverlos.

Cuando dos personas modifican la misma parte del código (por ejemplo, la misma línea o archivo), puede generarse un conflicto.

Visual Studio proporciona una interfaz visual que permite ver lado a lado las diferencias: tu versión, la versión remota, y elegir cómo unirlas (“merge”).

Es buena práctica revisar cuidadosamente los conflictos para que no se pierda funcionalidad o se introduzcan errores.

Para trabajar de forma óptima lo mejor es lo siguiente:

Trabajar en ramas (branches) separadas para cada nueva funcionalidad o cambio significativo, evitando tocar directamente la rama principal.

Hacer commits frecuentes, con mensajes claros que indiquen qué se cambió y por qué.

Antes de hacer merge a la rama principal (main, master, etc.), revisar los cambios, probar la funcionalidad integrada, y asegurarse de que no haya conflictos ni errores.

Mantener la sincronización activa: hacer pull regularmente para no acumular demasiados cambios divergentes.

Usar revisiones de código (code reviews) si el equipo lo permite, para que otro par de ojos revise antes de fusionar.