

# Guía de Colaboración con Git y GitHub en Visual Studio Code

## Introducción: Fundamentos de la Colaboración con Git en VS Code

### Propósito y Objetivos de Aprendizaje

Esta actividad está diseñada para simular un flujo de trabajo colaborativo real, tal como se practica en la industria del desarrollo de software. La idea de esta actividad es ver que lo que hemos hecho en la actividad anterior no es un conjunto aislado de comandos, sino parte de un proceso más amplio y estructurado que facilita la colaboración efectiva entre múltiples desarrolladores. Veremos que el proceso se puede hacer a través de una interfaz gráfica, en este caso, Visual Studio Code (VS Code).

Por tanto, todo el proceso de la actividad se llevará a cabo íntegramente dentro de **Visual Studio Code (VS Code)**, demostrando el poder de un **Entorno de Desarrollo Integrado (IDE)** para centralizar y optimizar las tareas de un desarrollador. Se aprenderá a transformar el editor de un simple procesador de texto a un centro de mando para la colaboración y el control de calidad.

### Requisitos Previos

Para llevar a cabo esta actividad, cada miembro del equipo debe cumplir con los siguientes requisitos:

1. **Cuenta de GitHub:** Cada estudiante debe tener una cuenta personal y activa en GitHub.
2. **Repositorio Compartido:** El equipo trabajará sobre un repositorio de GitHub llamado `pruebas` creado en la actividad anterior. Se asume que este repositorio ya ha sido creado y que todos los miembros del equipo han sido añadidos como colaboradores, otorgándoles permisos de escritura y existe una rama principal llamada `main` y una rama personal para cada miembro del equipo.

# Parte 1: Configuración del Entorno de Desarrollo Profesional

La preparación de un entorno de trabajo limpio, reproducible y adaptado a la tarea es la primera señal de un profesional. En lugar de instalar software directamente en el sistema operativo, se configurará una versión "portátil" de VS Code. Esto encapsula la aplicación y todas sus configuraciones en una única carpeta, permitiendo moverla entre ordenadores (por ejemplo, de un PC de casa a uno del laboratorio de la universidad) sin perder ninguna personalización y sin requerir permisos de administrador.

Dentro de esta instalación portátil, se creará un "**Perfil**" específico para **trabajar con Markdown**. Los perfiles permiten tener conjuntos aislados de extensiones y configuraciones para diferentes tipos de proyectos. Así, las herramientas para un proyecto de desarrollo web no interferirán con las de un proyecto de análisis de datos o, como en este caso, de documentación. Esta práctica de encapsulación y especialización del entorno es un pilar fundamental en los flujos de trabajo de DevOps y la ingeniería de software moderna.

## 1.1. Instalación de Visual Studio Code en Modo Portátil

Para asegurar que el entorno sea autónomo y no invasivo, se utilizará el modo portátil de VS Code.

### 1. Descargar la Versión Correcta:

Id a la [página oficial de descargas de Visual Studio Code](#). Es crucial que descarguen la versión en formato de archivo comprimido. En **Windows**, seleccionaremos la descarga **.zip**.



#### Importante

No utilizad los instaladores de sistema (User o System installers), ya que el modo portátil solo es compatible con estas versiones comprimidas.

### 2. Descomprimir y Configurar:

Una vez descargado el archivo, descomprimidlo en una ubicación de su elección (por ejemplo, en la unidad `C:\` o en una carpeta llamada `Herramientas`).

Descomprimid el zip en la carpeta recién creada (ej. `VSCode-win32-x64-[versión]`).

Dentro de esta carpeta, creen una nueva carpeta y hay que llamarla exactamente `data`.




#### Nota

La creación de esta carpeta **data** activa automáticamente el modo portátil. A partir de ahora, todas las configuraciones, extensiones, historial de sesión y preferencias se guardarán dentro de esta carpeta, haciendo que toda la aplicación sea autocontenida.

## 1.2. Creación de un Perfil de Trabajo Dedicado: "Markdown"

Un perfil de VS Code es un conjunto de personalizaciones que se puede activar o desactivar según la tarea. Esto mantiene el espacio de trabajo limpio y optimizado, cargando solo las herramientas necesarias para el proyecto actual.

1. **Iniciar VS Code:** Ejecuten el programa desde la carpeta que descomprimieron (ej. Code.exe).
2. **Acceder al Gestor de Perfiles:**
  - Vayan al menú **Archivo > Preferencias > Perfiles**.
  - Seleccionen la opción Crear Perfil.
3. **Configurar el Nuevo Perfil:**
  - En el cuadro de diálogo, se les pedirá que elijan una plantilla. Seleccionen Crear un perfil vacío (Create an Empty Profile) para empezar con una configuración limpia.
  - Asignen el nombre **Markdown** al nuevo perfil.
  - Puedes seleccionar un ícono representativo, como un **libro abierto** () , para facilitar la identificación visual del perfil.
  - Asegúrense de que todas las casillas de configuración (Settings, Keyboard Shortcuts, Extensions, etc.) estén marcadas para que el perfil sea completamente independiente.
  - Hagan clic en el botón Crear.
4. **Verificar el Perfil Activo:** Una vez creado, el nombre del perfil ("Markdown") aparecerá en la barra de título de VS Code y/o como una insignia en el ícono de engranaje (Gestionar) en la esquina inferior izquierda de la barra de actividades. Esto confirma que cualquier cambio o instalación de extensión **se aplicará únicamente a este perfil**.

## 1.3. Potenciando el Editor: Instalación de Extensiones Clave

Las extensiones transforman a VS Code de un editor de texto a un asistente de desarrollo activo. La siguiente selección de herramientas está cuidadosamente curada para proporcionar una experiencia completa, abarcando desde la localización y la calidad del texto hasta la integración con el flujo de trabajo de Git y GitHub.

Para instalar cada extensión, con el perfil "Markdown" activo:

1. Abran la vista de **Extensiones** en la barra de actividades (el icono de los cuatro cuadrados) o presionen Ctrl+Shift+X.

2. En la barra de búsqueda, escriban el nombre o el identificador único de la extensión.
3. Hagan clic en Instalar en la extensión correcta.

A continuación se presenta una tabla con las extensiones a instalar y su propósito. Instalen cada una de ellas.

<b>Nombre de la Extensión</b>	<b>Identificador Único</b>	<b>Editor</b>	<b>Propósito Principal</b>
Spanish Language Pack	MS-CEINTL.vscode-language-pack-es	Microsoft	Traduce la interfaz de VS Code al español.
Code Spell Checker	streetsidesoftware.code-spell-checker	Street Side Software	Corrector ortográfico genérico para código y texto.
Spanish - Code Spell Checker	streetsidesoftware.code-spell-checker-spanish	Street Side Software	Añade el diccionario de español al corrector ortográfico.
open in browser	techer.open-in-browser	TechER	Permite abrir archivos HTML en el navegador (útil para vistas previas).
Markdown All in One	yzhang.markdown-all-in-one	Yu Zhang	Proporciona atajos y herramientas esenciales para Markdown. <sup>12</sup>
Markdown Preview Enhanced	shd101wyy.markdown-preview-enhanced	Yiyi Wang	Ofrece una previsualización avanzada y personalizable de Markdown. <sup>13</sup>
markdownlint	DavidAnson.vscode-markdownlint	David Anson	Analiza el código Markdown en busca de errores de estilo y consistencia. <sup>14</sup>
PlantUML	jebbs.plantuml	jebbs	Proporciona resaltado de sintaxis para diagramas PlantUML en bloques de código. <sup>15</sup>
GitGraph	mhutchie.git-graph	mhutchie	Visualiza el historial de Git como un grafo interactivo. <sup>17</sup>

Nombre de la Extensión	Identificador Único	Editor	Propósito Principal
Visual Studio Keymap	ms-vscode.vs-keybindings	Microsoft	Adapta los atajos de teclado a los de Visual Studio tradicional.
GitHub Pull Requests and Issues	GitHub.vscode-pull-request-github	GitHub	Integra la gestión de Pull Requests e Issues de GitHub en VS Code.18

Una vez instalada la extensión Spanish Language Pack, es posible que VS Code les pida reiniciar para aplicar el cambio de idioma. Acepten y continúen. La combinación de Code Spell Checker y markdownlint crea una especie de "control de calidad" local que detecta errores antes de que el código sea siquiera guardado, una práctica esencial para mantener la calidad del proyecto.

## Parte 2: El Ciclo de Desarrollo Colaborativo: De la Sincronización a la Contribución

El núcleo de la colaboración efectiva en Git se basa en un ciclo disciplinado conocido como el **Flujo de Trabajo de Ramas de Característica** (*Feature Branch Workflow*). Cada nueva tarea o cambio se desarrolla en una rama aislada para no desestabilizar la base de código principal (main) esta rama puede ser perfectamente **la nuestra como miembro del equipo**.



Recuerda que ...

Un paso fundamental, y a menudo olvidado por los principiantes, es actualizar la rama de trabajo personal con los últimos cambios de main *antes* de empezar a programar. Esta práctica de **evitación proactiva de conflictos** es mucho más eficiente que resolver complejas fusiones más adelante. Es análogo a consultar la última versión del plano maestro de un edificio antes de empezar a construir una nueva pared.


### 2.1. Preparando el Espacio de Trabajo

1. **Abrir el Repositorio:** Abrid la paleta de comandos con `Ctrl+Shift+P` escriban **Git: Clonar**. Presionen Enter.
2. Seleccionen **Clonar desde GitHub** y, si es la primera vez, sigan los pasos para autenticarse con su cuenta de GitHub.

3. Busquen y seleccionen el repositorio **pruebas** de la lista. Elijan una carpeta en su ordenador donde se guardará la copia local del proyecto. VS Code abrirá automáticamente el proyecto una vez clonado.
4. **Verificar la Rama Actual:** En la **esquina inferior izquierda** de la ventana de VS Code, en la barra de estado, verán el nombre de la rama actual, que por defecto será **main**.
5. **Cambiar a la rama Personal:** Puesto que ya creamos la rama personal en la actividad anterior, ahora debemos cambiar a ella. Hagan clic en el nombre de la rama (main) en la barra de estado. En el menú que aparece, seleccionen tu rama personal (ej. **ana**).

## 2.2. Sincronización: La Clave para Evitar Conflictos

Antes de escribir una sola línea de código, es vital asegurarse de que la rama personal contiene todos los cambios que otros compañeros ya hayan integrado en main.

1. **Sincronizar Cambios:** Con tu rama personal activa, busquen el botón de sincronización en la barra de estado (suele tener un icono de flechas circulares  ) o vayan a la vista de Control de código fuente ( **Ctrl+Shift+G** ), hagan clic en los tres puntos (...) y seleccionar la opción **"Pull", "Push" > Extraer de** . En el menú que aparece, elijan **origin/main** . Esto traerá todos los commits de main y los fusionará en su rama personal.
2. **Visualizar el Historial:** Para comprender lo que acaba de ocurrir, abran la extensión **GitGraph**.

## 2.3. Aportando Contenido: Edición, Commit y Push

Ahora que la rama está sincronizada, es el momento de realizar los cambios.

1. **Añadir Contenido:** Cada estudiante debe abrir su archivo Markdown personal (ej. **ana\_garcia.md** ) y añadir nuevo contenido, como un párrafo describiendo sus hobbies o una lista de sus asignaturas favoritas.
2. **Preparar los Cambios (Stage):** Vayan a la vista de **Control de código fuente** ( **Ctrl+Shift+G** ). Verán que su archivo modificado aparece en la sección "Cambios", marcado con una **M** (**Modified**). Hagan clic en el icono ( **+** ) junto al nombre del archivo para **"prepararlo"**. Esta acción, equivalente a **git add** , le dice a Git que este cambio específico debe ser incluido en la próxima **"instantánea"** (commit) del proyecto.
3. **Realizar la Confirmación (Commit):** En el cuadro de texto en la parte superior de la vista de Control de código fuente, escriban un **mensaje de commit claro y descriptivo**.



### Importante

Un buen mensaje explica *qué* se cambió y *por qué*. Por ejemplo:

"Feat: Añade sección de hobbies al perfil de Ana García ". Evitar mensajes genéricos como "cambios" o "actualización".

4. Hagan clic en el icono de la marca de verificación (✓) para confirmar los cambios. Este commit ahora existe en su repositorio local.
5. **Empujar los Cambios (Push):** Para compartir su commit con el resto del equipo, deben enviarlo al repositorio remoto en GitHub. Hagan clic en el botón Publicar cambios o Sincronizar cambios en la barra de estado. Esto subirá su rama y su nuevo commit a GitHub.

## Parte 3: Revisión por Pares: El Proceso de Pull Request

Una *Pull Request* (PR) es el corazón del flujo de trabajo colaborativo en GitHub. No es solo una solicitud para fusionar código; es el inicio de una conversación estructurada sobre la calidad y la idoneidad de los cambios propuestos. La extensión GitHub Pull Requests and Issues integra este diálogo directamente en VS Code, transformando el editor en un centro de revisión de código.<sup>19</sup> Este proceso fomenta una cultura de responsabilidad compartida y mejora continua, habilidades blandas tan cruciales como las habilidades técnicas en el desarrollo de software.

### 3.1. Proponiendo Cambios: Creación de una Pull Request (PR)

Una vez que han subido sus cambios a GitHub, el siguiente paso es proponer su integración en la rama main.

1. **Abrir la Vista de GitHub:** Vayan a la vista de **GitHub** en la barra de actividades (el icono de octocat).
2. **Crear la Pull Request:** En la parte superior del panel, verán un botón para **Crear Pull Request**. Hagan clic en él.
3. **Configurar la PR:** Se abrirá una nueva pestaña en VS Code para configurar la solicitud. Rellenen los siguientes campos:
  - **Desde:** Asegúrense de que **tu rama personal** (ej. `ana`) está seleccionada.
  - **Hacia:** Seleccionen la rama `main` como **destino**.
  - **Título:** Escriban un **título descriptivo** para la PR, que resuma el cambio. Por ejemplo: `Añadir documentación del perfil de Ana García`.
  - **Descripción:** Proporcionen más detalles sobre los cambios realizados.
  - **Revisores (Reviewers):** En el panel derecho o en la descripción, asignen a otro compañero del equipo como revisor. Para ello, escriban `@` seguido de su nombre de usuario de GitHub (ej. `@juanperez`). El autocompletado les ayudará a encontrarlo.



## Importante

En este punto cada compañero elegirá a otro miembro del equipo como revisor, asegurando que cada PR sea revisada por alguien diferente. De tal manera que todos los compañeros hagan un pull request y revisen el de otro compañero.

4. **Enviar la Solicitud:** Hagan clic en el botón Crear en la esquina inferior derecha. La Pull Request ahora está abierta en GitHub y el revisor asignado recibirá una notificación.

## 3.2. Asegurando la Calidad: Revisión, Aprobación y Fusión (Merge)

Ahora, el proceso pasa al estudiante que fue designado como **revisor**.

1. **Localizar la PR Asignada:** En su propio VS Code, el revisor verá la Pull Request asignada en la categoría "**Esperando mi revisión**" (Waiting For My Review) dentro de la vista de la extensión de GitHub.
2. **Iniciar la Revisión:** Hagan clic en la PR para abrirla. Verán una vista de descripción general. Para ver los cambios en el código, hagan clic en el nombre del archivo modificado en la sección "**Cambios en archivos**" (Files Changed). Esto abrirá una vista diff que muestra las líneas añadidas (en verde) y eliminadas (en rojo).
3. **Paso Crítico - Probar los Cambios (Checkout):** La revisión de código más efectiva no es solo visual. En la vista de descripción de la PR, hay un botón **Revisar** (Checkout). Al hacer clic, VS Code descargará temporalmente el estado del código de la PR, permitiendo al revisor compilarlo, ejecutarlo y probarlo localmente para asegurarse de que todo funciona como se espera. Esta es una práctica profesional estándar para validar la funcionalidad, no solo la sintaxis. En nuestro caso solo es texto en Markdown, pero en proyectos reales puede implicar ejecutar pruebas automatizadas o revisar la interfaz de usuario.
4. **Aprobar y Fusionar:** Una vez que el revisor ha validado que los cambios son correctos y no introducen problemas, debe volver a la vista de descripción de la PR.
  - Dejen un comentario de aprobación, como "¡Buen trabajo! Todo parece correcto."
  - Hagan clic en el botón **Aprobar** (Approve).
  - Finalmente, hagan clic en el botón **Fusionar Pull Request** (Merge Pull Request). Esta acción integra definitivamente los cambios de la rama del compañero en la rama main.
5. **Limpieza del Repositorio:** Después de la fusión, GitHub (y la extensión de VS Code) **ofrecerá la opción de eliminar la rama de característica**, ya que su propósito ha sido cumplido. Es una buena práctica **eliminar estas ramas para mantener el repositorio ordenado**.



# Conclusión: Integrando Prácticas Profesionales en tu Flujo de Trabajo

## Resumen de Habilidades Adquiridas

A lo largo de esta actividad, el equipo ha practicado un ciclo de desarrollo completo que va más allá de la simple escritura de código. Han aprendido y aplicado conceptos fundamentales para la ingeniería de software moderna:

- **Configuración de un entorno de desarrollo profesional:** Crearon un entorno de trabajo aislado y reproducible mediante el modo portátil y los perfiles de VS Code.
- **Flujo de trabajo en ramas (Feature Branch Workflow):** Aislaron los cambios en ramas personales para no interferir con el trabajo de los demás.
- **Sincronización proactiva:** Aprendieron la importancia de actualizar sus ramas desde main para prevenir conflictos.
- **Commits atómicos y descriptivos:** Practicaron la creación de "instantáneas (snapshots)" del proyecto con mensajes claros que construyen un historial legible.
- **Ciclo de Pull Request:** Dominaron el proceso de propuesta, revisión por pares, aprobación y fusión de cambios, todo ello desde la "comodidad" de su editor de código.

## Próximos Pasos y Relevancia Profesional

El flujo de trabajo que han practicado es una versión simplificada de los procesos que utilizan miles de equipos de desarrollo de software en todo el mundo, desde startups hasta grandes corporaciones tecnológicas. Estas **no son solo "buenas prácticas", sino habilidades esenciales** que se esperan de cualquier desarrollador profesional.

Deberéis adoptar este ciclo de:

1. Trabajar en mi rama personal.
2. Sincronizar con origin/main.
3. Hacer commits claros y atómicos.
4. Crear Pull Requests para revisión.

Para todos vuestros proyectos en grupo. Construir estos hábitos desde el principio no solo mejorará la calidad de su trabajo y facilitará la colaboración, sino que también les proporcionará una ventaja significativa al entrar en el mundo profesional. Han dado el primer paso para pasar de ser estudiantes que escriben código a ser ingenieros que construyen software de manera colaborativa y sostenible.