

Informe de Actividad Práctica

Registro de Comandos y Commits en GitHub

Ignacio Ramírez
Antonia Montecinos
Cristian Vergara

Docente: Michael Miranda
Octubre de 2025

Semana 2

Índice

1. Introducción	2
2. Desarrollo de la Actividad y Evidencias	2
2.1. Creación y Clonación del Repositorio	2
2.2. Registro de Comandos Git y Commits	3
2.3. Prueba de GitHub Copilot	4
3. Conclusiones	5

1. Introducción

Este informe documenta la actividad práctica de la Semana 2, centrada en el uso fundamental de Git y GitHub como herramientas para el control de versiones y la gestión de proyectos. El objetivo fue aplicar los conceptos básicos de Big Data a la práctica del desarrollo, creando un repositorio para centralizar el trabajo del curso y familiarizarse con el flujo de trabajo estándar de clonación, confirmación de cambios y subida al repositorio remoto.

Adicionalmente, se exploró la herramienta de asistencia por IA, GitHub Copilot, para evaluar su capacidad de agilizar la codificación. Este laboratorio es esencial para establecer una metodología de trabajo organizada y versionada.

2. Desarrollo de la Actividad y Evidencias

El desarrollo de la actividad se dividió en tres etapas principales: la creación y clonación del repositorio, el registro de los primeros commits y la prueba funcional de GitHub Copilot.

2.1. Creación y Clonación del Repositorio

El primer paso fue crear un nuevo repositorio público en GitHub, llamado `INFB6052-Herramientas-para-Ciencia-de-Datos`. Se configuró con un nombre descriptivo, una breve descripción y se inicializó con un archivo `README.md`.

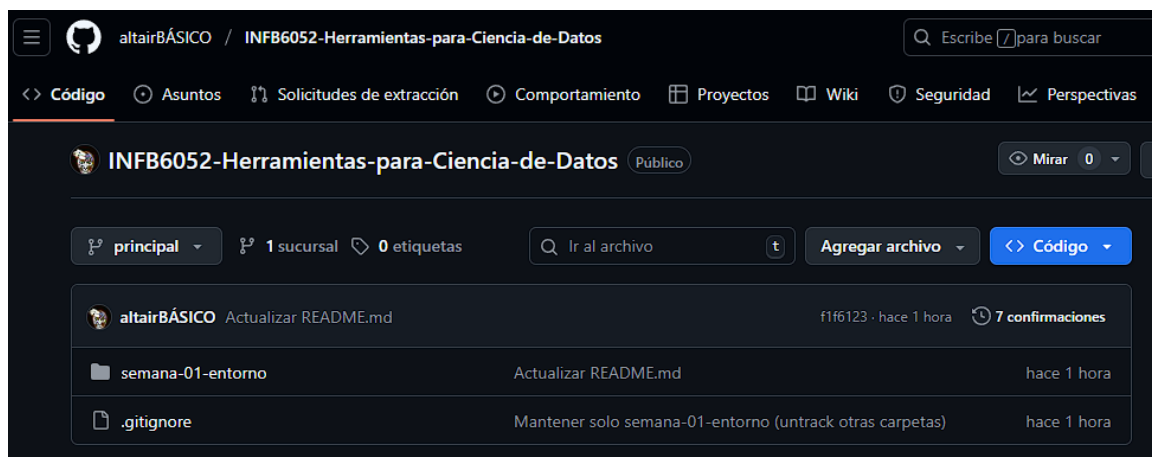


Figura 1: Página principal del repositorio recién creado en GitHub.

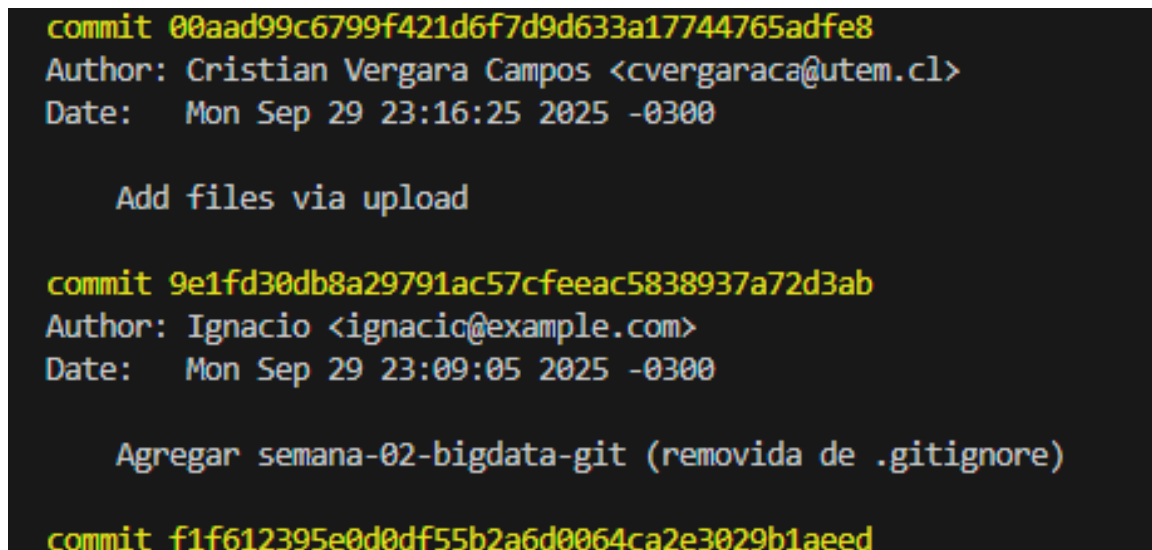
Posteriormente, el repositorio remoto fue clonado a nuestro entorno local utilizando el comando `git clone` con la URL HTTPS proporcionada por GitHub.

2.2. Registro de Comandos Git y Commits

Una vez clonado el repositorio, se procedió a realizar las primeras modificaciones. Se creó una estructura de carpetas inicial y se añadió un archivo `.gitignore` para excluir los archivos del entorno virtual. Los comandos utilizados en este flujo de trabajo fueron:

- **git status:** Para verificar el estado de los archivos (modificados, nuevos, etc.).
- **git add .:** Para añadir todos los archivos nuevos y modificados al área de "staging".
- **git commit -m "Mensaje descriptivo":** Para confirmar los cambios con un mensaje claro que explica la modificación.
- **git push origin main:** Para subir los commits locales al repositorio remoto en GitHub.

La Figura 2 muestra el historial de los commits iniciales realizados, evidenciando la progresión del trabajo y el uso de mensajes de commit descriptivos siguiendo buenas prácticas.



```
commit 00aad99c6799f421d6f7d9d633a17744765adfe8
Author: Cristian Vergara Campos <cvergaraca@utem.cl>
Date:   Mon Sep 29 23:16:25 2025 -0300

    Add files via upload

commit 9e1fd30db8a29791ac57cfeeac5838937a72d3ab
Author: Ignacio <ignacio@example.com>
Date:   Mon Sep 29 23:09:05 2025 -0300

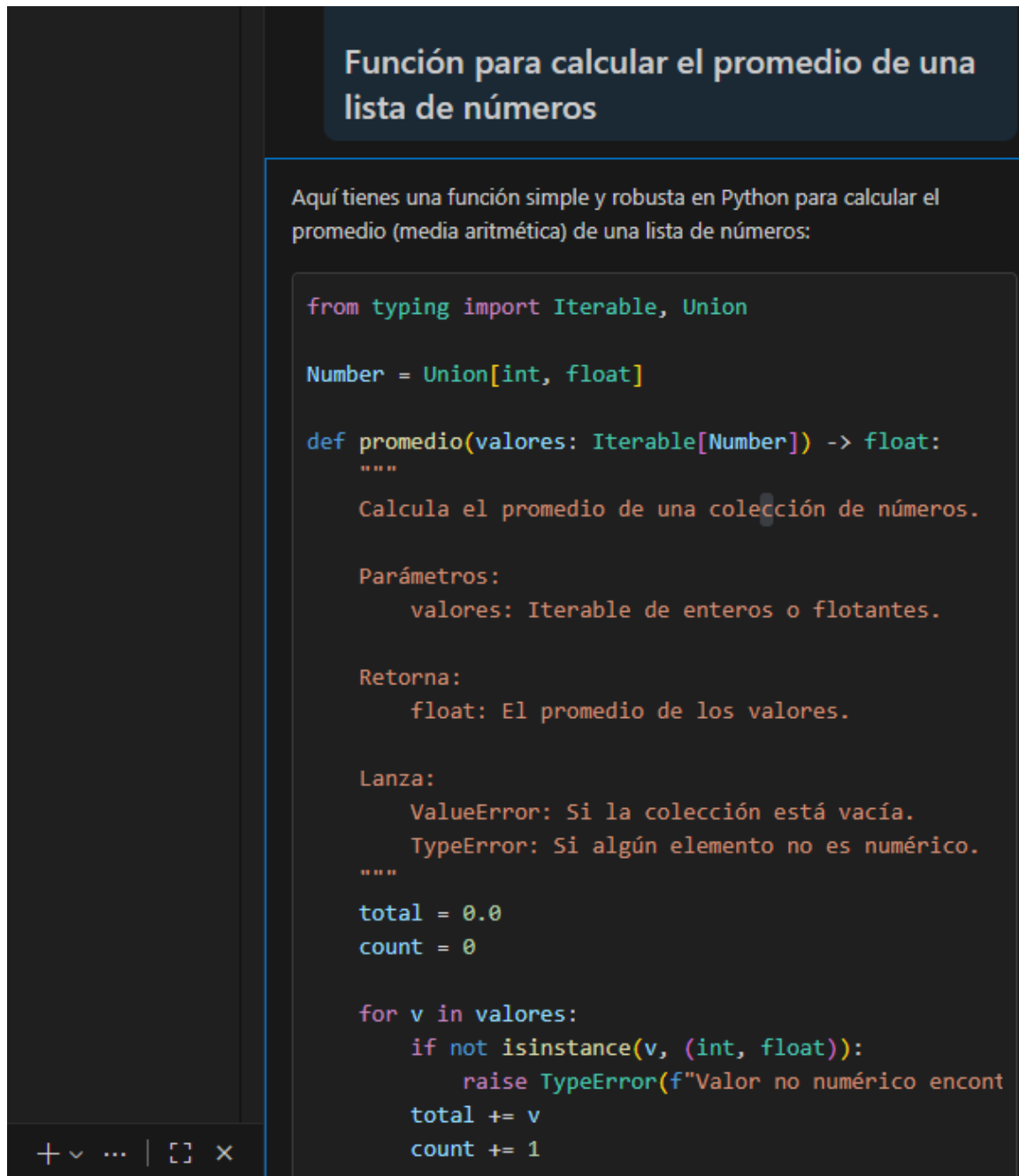
    Agregar semana-02-bigdata-git (removida de .gitignore)

commit f1f612395e0d0df55b2a6d0064ca2e3029b1aeed
```

Figura 2: Salida del comando `git log -oneline` mostrando los primeros commits del proyecto.

2.3. Prueba de GitHub Copilot

Finalmente, se realizó una prueba funcional de la extensión GitHub Copilot. Dentro de un archivo Python, se escribió un comentario descriptivo: `# Función para calcular el promedio de una lista de números`. Inmediatamente, Copilot sugirió un bloque de código completo y funcional que cumplía con lo solicitado, demostrando su utilidad como herramienta de productividad.

The image shows a dark-themed code editor interface. At the top, a blue header bar contains the text "Función para calcular el promedio de una lista de números" in white. Below this, a light blue box contains the text "Aquí tienes una función simple y robusta en Python para calcular el promedio (media aritmética) de una lista de números:". The main area of the editor displays a Python function definition for "promedio". The function uses type hints from the "typing" module, including "Iterable" and "Union". It includes a docstring with a description, parameter information, return type, and exceptions. The function logic initializes "total" and "count", then iterates over the input list, checking for numeric values and calculating the average. At the bottom left, a toolbar shows icons for adding, removing, and navigating code blocks.

```
from typing import Iterable, Union

Number = Union[int, float]

def promedio(valores: Iterable[Number]) -> float:
    """
    Calcula el promedio de una colección de números.

    Parámetros:
        valores: Iterable de enteros o flotantes.

    Retorna:
        float: El promedio de los valores.

    Lanza:
        ValueError: Si la colección está vacía.
        TypeError: Si algún elemento no es numérico.
    """
    total = 0.0
    count = 0

    for v in valores:
        if not isinstance(v, (int, float)):
            raise TypeError(f"Valor no numérico encont
        total += v
        count += 1
```

Figura 3: Sugerencia de código generada por GitHub Copilot a partir de un comentario en Python.

3. Conclusiones

La actividad de la Semana 2 se completó con éxito, logrando establecer un flujo de trabajo básico pero fundamental con Git y GitHub. La creación del repositorio central y la práctica de los comandos `add`, `commit` y `push` solidifican las bases para el control de versiones que se aplicará en todos los proyectos futuros.

La prueba de GitHub Copilot resultó impresionante, demostrando ser una herramienta potente para acelerar el desarrollo y reducir errores de sintaxis. La familiarización con estas herramientas es un paso crucial hacia la adopción de prácticas de desarrollo de software modernas en el campo de la ciencia de datos.