

Nombre del bloque PROGRAMACIÓN II

Año de cursada 1º AÑO

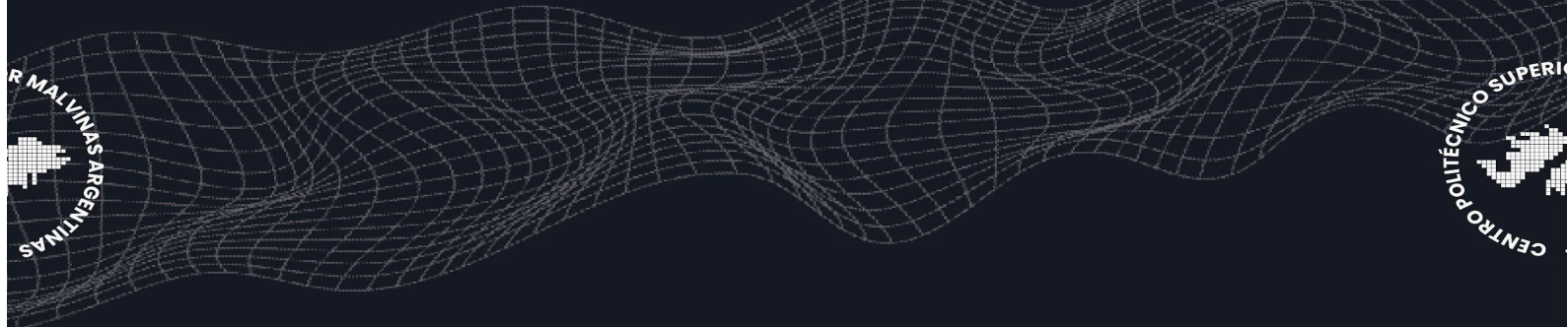
Clase N° 3: Trabajando con GitHub y VsCode



Contenido:

En la clase de hoy trabajaremos los siguientes temas:

- Gestión de archivos en GitHub y Visual Studio.
- Colaboración y Sincronización de repositorios entre Visual Studio y GitHub.
- Creación y eliminación de ramas.
- Gestión y utilización de ramas.



Presentación:

Bienvenidos a la clase III de Programación II 🖐️. En la clase anterior, exploramos los conceptos básicos de Git, GitHub y la integración con Visual Studio Code. Hoy, continuaremos nuestra inmersión en el mundo del control de versiones y la colaboración en el desarrollo de software.

Repaso rápido:

- En la clase anterior, aprendimos sobre Git, un sistema de control de versiones distribuido.
- Exploramos GitHub, una plataforma para alojar proyectos de software y colaborar con otros desarrolladores.
- También examinamos cómo integrar Git en Visual Studio Code para mejorar nuestra experiencia de desarrollo.

En la clase de hoy vamos a estar desarrollando los siguientes temas:

Gestión de archivos en GitHub y Visual Studio:

Veremos la importancia de mantener una estructura organizada de archivos en nuestros repositorios tanto en GitHub como en Visual Studio. Veremos cómo agregar, modificar y eliminar archivos de manera efectiva.

Colaboración y Sincronización de repositorios:

Ahora nos adentraremos en aspectos más avanzados de la colaboración y sincronización de repositorios entre Visual Studio y GitHub. Veremos cómo manejar conflictos, realizar fusiones y asegurar una colaboración fluida entre equipos.

Creación y eliminación de ramas:



Las ramas son una característica esencial en Git que nos permite trabajar en paralelo en diferentes partes de nuestro proyecto. Revisaremos cómo crear y eliminar ramas de manera adecuada para mantener un flujo de trabajo eficiente.

Gestión y utilización de ramas:

Profundizaremos en la gestión y utilización de ramas, explorando estrategias para fusionar ramas, resolver conflictos y mantener un historial limpio de cambios en nuestro código.

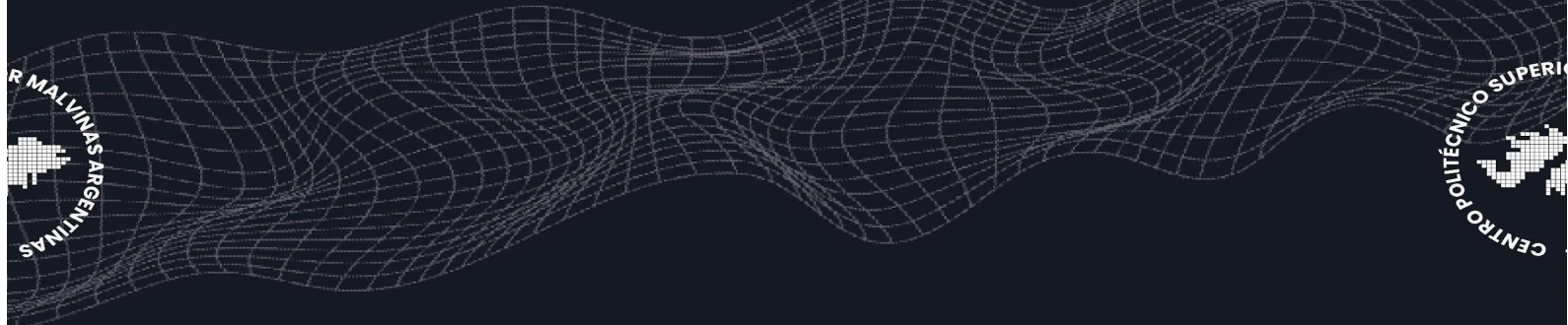
Práctica en vivo:

Para consolidar lo aprendido, realizaremos ejercicios prácticos en tiempo real. Trabajaremos en equipos, colaborando en repositorios compartidos en GitHub y utilizando Visual Studio Code para realizar cambios y gestionar versiones.

Objetivos de Aprendizaje:

- Reforzar los conocimientos adquiridos sobre Git, GitHub y la integración con Visual Studio Code.
- Dominar técnicas avanzadas de colaboración y sincronización de repositorios.
- Desarrollar habilidades para gestionar eficientemente ramas y resolver conflictos en el desarrollo de software colaborativo.

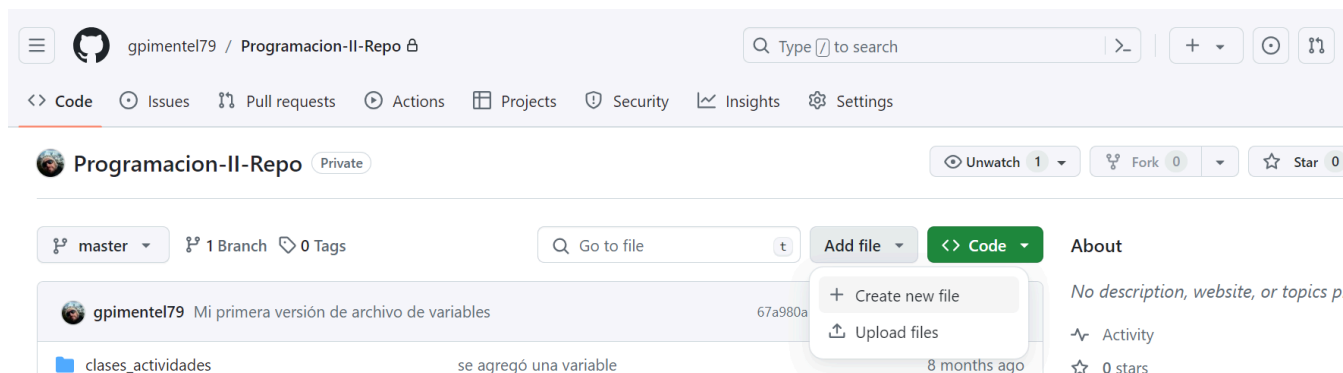
Los animo a participar activamente y hacer preguntas. Estoy aquí para ayudarlos a entender y aprovechar al máximo estos conceptos. ¡Sigamos aprendiendo juntos y disfrutemos de esta clase!



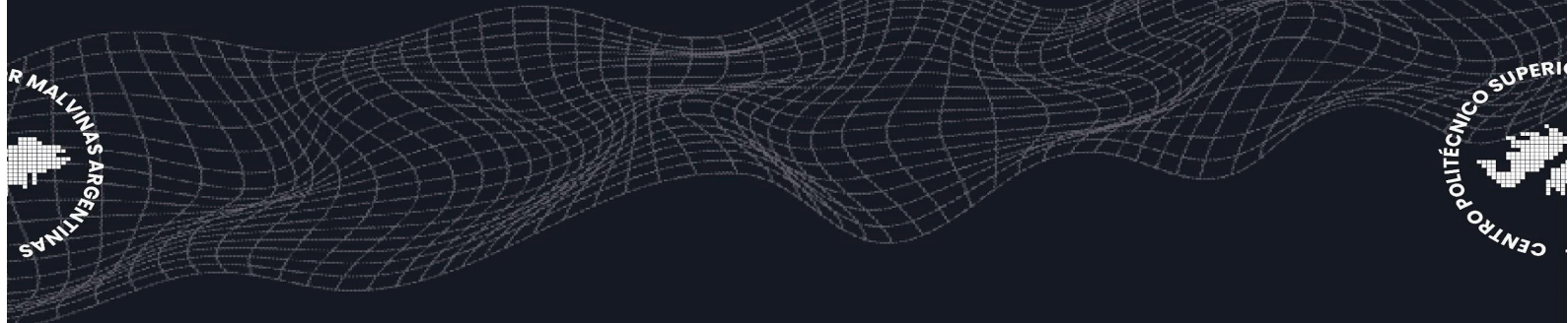
Desarrollo y Actividades:

Creación y edición de archivos en GitHub

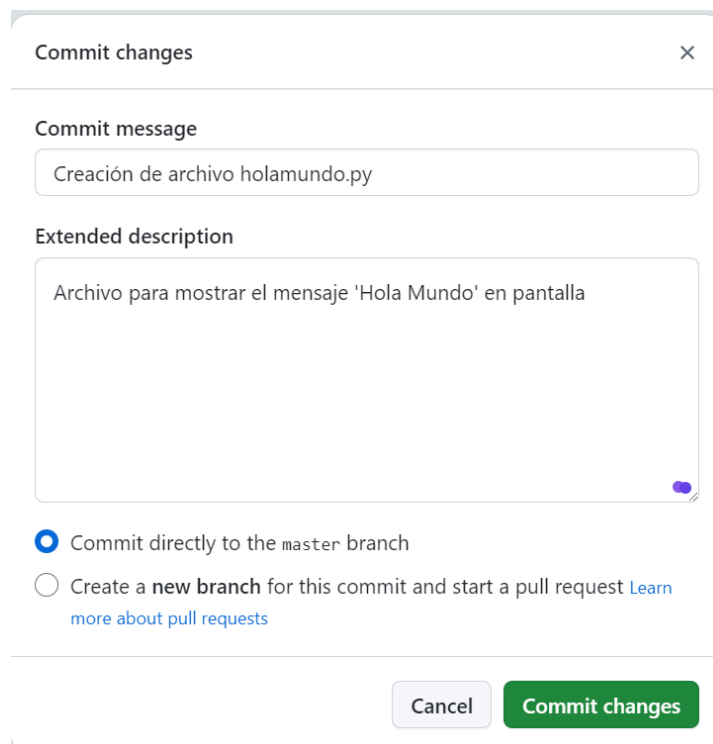
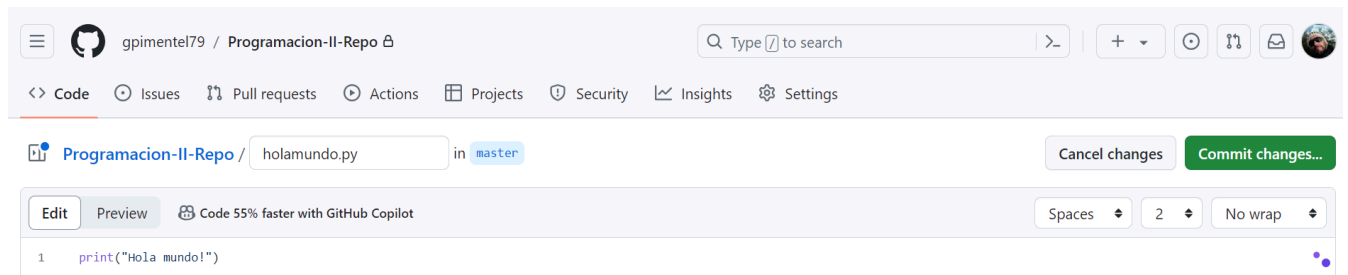
GitHub nos permite crear archivos y editarlos directamente desde su página oficial. Vamos a ver cómo crear archivos y editarlos directamente en GitHub a través de nuestra cuenta. Abrimos el repositorio que ya tenemos creado en GitHub y veamos cómo podemos crear un archivo cualquiera directamente desde la propia plataforma. En mi repositorio Programacion-II-Repo voy a crear un archivo llamado **holamundo.py**. Para ello, en la vista principal del repositorio, hacemos clic sobre el botón Add file y a continuación Create new file (crear nuevo archivo).



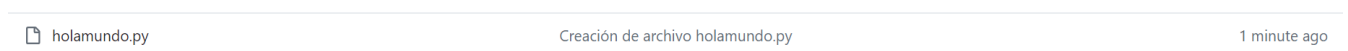
Es importante que no solo indiquemos el nombre del archivo, sino también su extensión. Por ejemplo, 'holamundo.py'. Y escribimos el contenido del archivo. A continuación, debemos anotar la creación de este archivo. En Git, estas anotaciones las realizamos mediante una acción llamada '**commit**'. Un '**commit**' no es más que anotar un cambio en nuestro repositorio. De este modo, Git puede almacenar el historial de modificaciones que se han ido realizando a lo largo del tiempo. Dado que este historial puede ser revisado, debemos incluir información relevante relativa a este cambio. Así que añadimos un título. Por ejemplo, "Creación de archivo 'Hola Mundo'". Y una descripción un poco más larga. Por ejemplo, "Archivo para mostrar el mensaje 'Hola Mundo' en



pantalla". Y por el momento vamos a lanzar el 'commit' directamente sobre la rama principal. Por tanto, hacemos clic sobre el botón Commit changes.



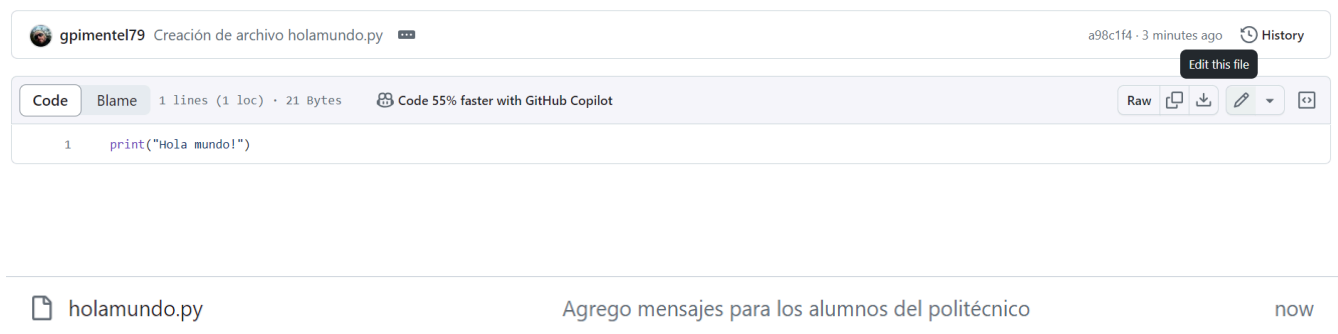
Como podemos ver, el archivo 'holaMundo.py' se ha creado correctamente en nuestro repositorio remoto. Además, podemos apreciar que ha sido la última acción que un usuario ha realizado sobre este repositorio y que dicha acción se produjo hace 1 minuto.




Para editar este o cualquier otro archivo desde GitHub, solo tenemos que hacer clic sobre

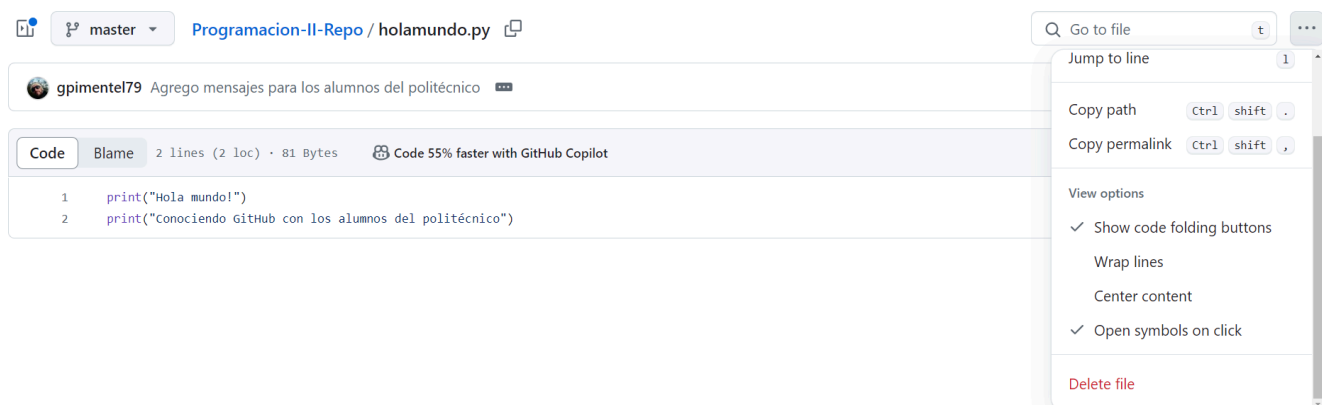
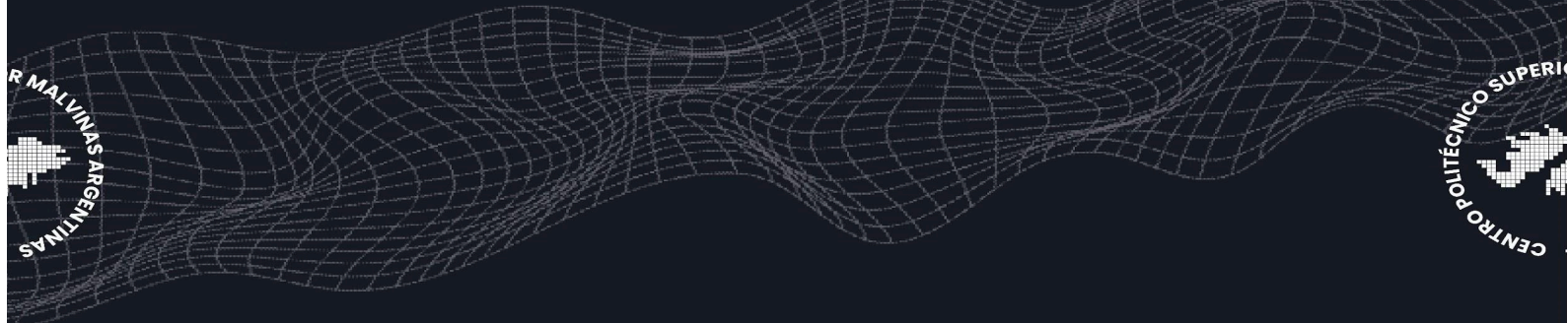


el fichero que deseamos modificar. A continuación, hacemos clic sobre el botón que contiene un lápiz y repetimos el mismo proceso que acabamos de realizar para la creación. Cambiamos lo que necesitemos y hacemos 'commit'. Por supuesto, indicamos el título y también añadimos una descripción. Y, como podemos apreciar, el cambio se ha realizado exitosamente. No solo lo vemos en el contenido del archivo, sino también en la visión general del repositorio.



Eliminación de archivos en GitHub

GitHub nos permite eliminar archivos directamente desde su página oficial. En Para eliminar un archivo cualquiera de nuestro repositorio remoto a través de GitHub, accedemos al repositorio en cuestión que aloja el archivo y a continuación hacemos clic sobre el título del archivo que deseamos eliminar. En mi caso va a ser 'holamundo.py'. Esto nos llevará a una vista que muestra el contenido del mismo con diferentes opciones. Entre ellas encontramos arriba a la derecha un icono con tres puntos en horizontal  hacemos click en él y se nos despliegan más acciones que podemos realizar sobre el archivo, nos desplazamos hacia el final y veremos la opción de eliminar el archivo.

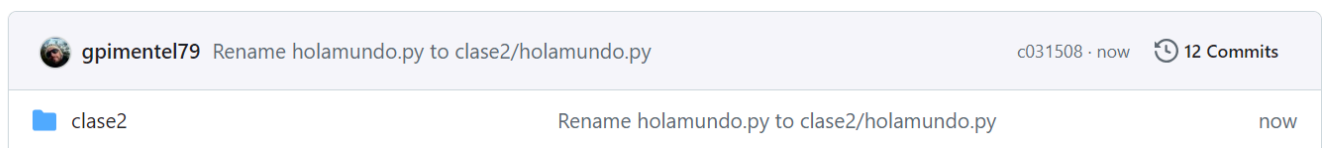


Hacemos clic sobre la opción y a continuación debemos anotar el cambio para que la eliminación se haga efectiva. Escribimos un título descriptivo y pulsamos sobre el botón verde Commit changes. Automáticamente seremos redirigidos a la página principal del repositorio con el que estamos trabajando. Y, como podemos observar, el archivo ya no se encuentra disponible.

Mover archivos en GitHub

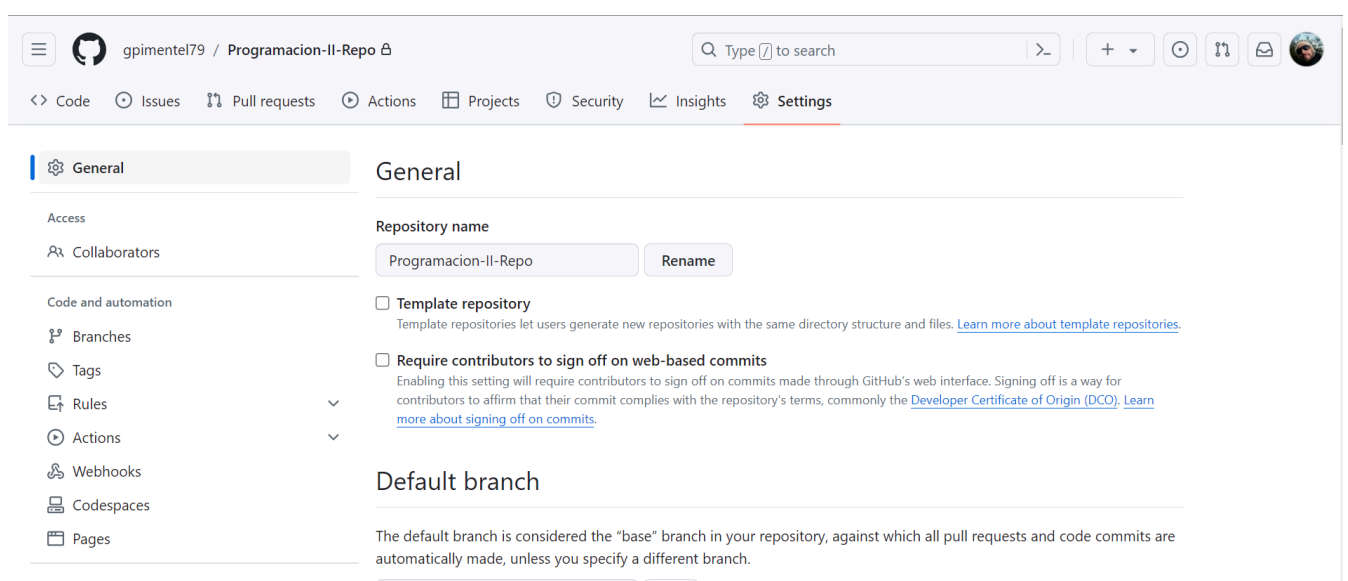
¿Qué ocurre cuando movemos un archivo en GitHub? Dado que en el futuro dispondremos de más ejercicios en este repositorio, tiene sentido alojarlos en diferentes carpetas y mostrarlos de un modo más organizado. Así que vamos a ver cómo mover un archivo a una carpeta llamada /clase2. Para ello hacemos clic sobre el título del archivo que deseamos mover y a continuación hacemos clic sobre el botón de editar. En el cuadro de texto donde podemos modificar el nombre del archivo vamos a escribir al comienzo del mismo el nombre de la carpeta a la que deseamos mover el fichero, y entre el nombre de la carpeta y el nombre del programa escribimos una barra inclinada hacia la derecha /. Automáticamente se creará una carpeta y en su interior se alojará este archivo. Por supuesto, debemos anotar este cambio en el repositorio. Así que nos dirigimos a la parte inferior de esta página, escribimos un título adecuado

relativo a la modificación que estamos realizando. Si ahora accedemos a la vista principal del repositorio, podremos comprobar que la carpeta se ha creado correctamente y en su interior se encuentra ahora el archivo en cuestión.

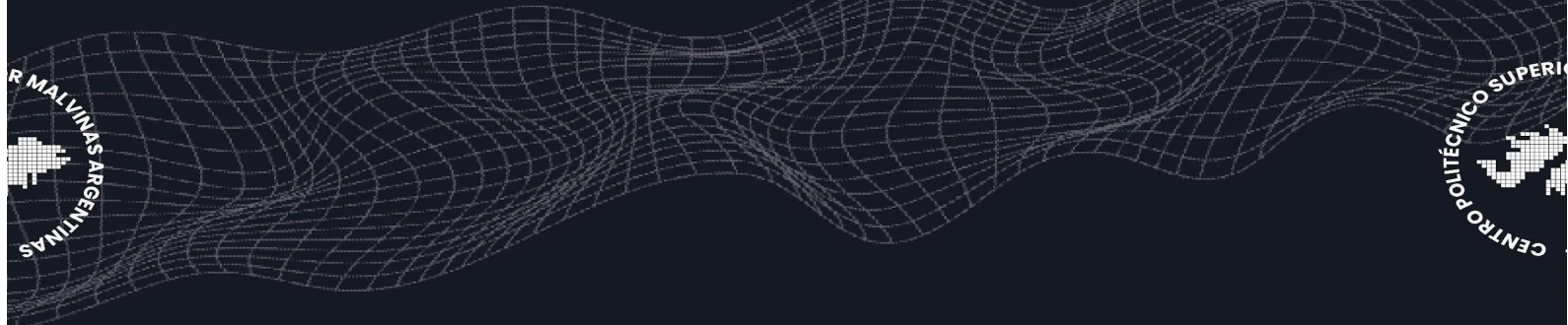


Añadir colaboradores a un repositorio en GitHub

La fortaleza de Git y GitHub se encuentra en la colaboración para el desarrollo de software. Para añadir colaboradores a un repositorio remoto al que pertenecemos alojado en GitHub, accedemos a la URL del repositorio en cuestión y, habiendo iniciado sesión, hacemos clic en Settings y a continuación seleccionamos la opción Collaborators.



Probablemente seamos redirigidos a una vista de confirmación de acceso en la

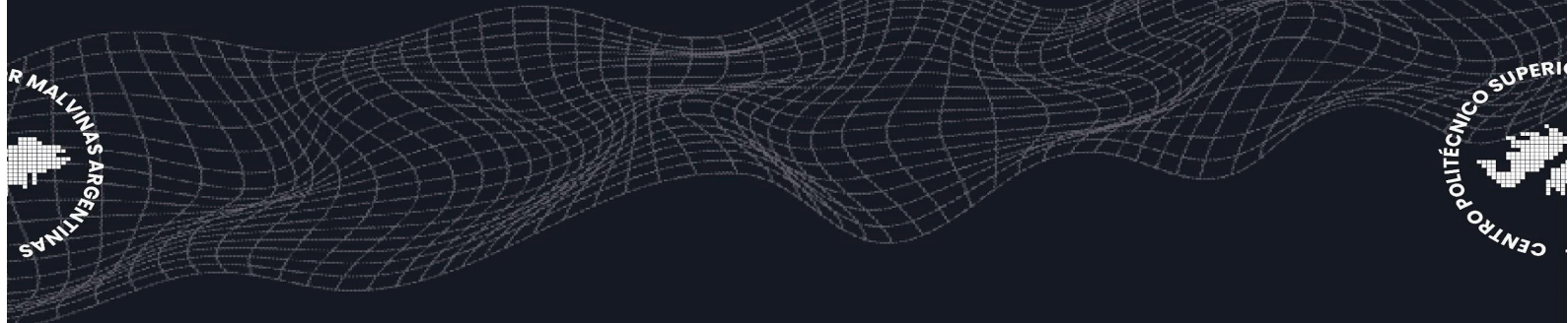


que puede que nos pidan, por cuestiones de seguridad, la introducción de un código de activación que recibiremos por mensaje de texto. Vamos a pedir que nos envíen ese mensaje de texto y lo introducimos en este cuadro de texto. Tras introducir dicho código, ahora sí, podremos gestionar el acceso de colaboradores al repositorio. Hacemos clic sobre el botón Add people y a continuación introducimos el nombre de usuario o el correo electrónico de la persona que deseamos añadir. Tras localizar al usuario en cuestión, lo seleccionamos y a continuación hacemos clic sobre el botón Add (nombre de usuario) to this repository. En estos momentos, nuestro colaborador dispondrá de siete días para aceptar nuestra invitación y convertirse en colaborador de nuestro repositorio.

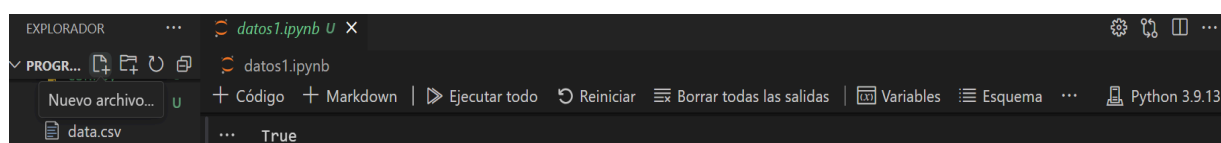
The screenshot shows the GitHub repository settings interface. On the left, a sidebar contains a list of settings categories: General, Access, Collaborators (highlighted), Code and automation, Branches, Tags, Rules, Actions, Webhooks, Codespaces, Pages, Security, Code security and analysis, Deploy keys, and Secrets and variables. The main content area is titled 'Who has access' and displays two selectable options: 'PRIVATE REPOSITORY' (which is selected and shows a lock icon) and 'DIRECT ACCESS' (which shows an open lock icon). Below these options, the 'Manage access' section is visible, featuring a message that reads 'You haven't invited any collaborators yet' accompanied by an icon of a person with a lock. A prominent green button labeled 'Add people' is positioned at the bottom of this section.

Creación y edición de archivos en la carpeta de un repositorio local a través de Visual Studio

Visual Studio nos permite crear archivos en nuestro repositorio local y también editarlos posteriormente. Dado que ya disponemos de un repositorio local en



nuestro equipo y estamos trabajando desde Visual Studio en la carpeta que lo contiene, podemos realizar cambios en este repositorio local. Para crear un nuevo archivo, arriba a la izquierda, se encuentra la carpeta que contiene el repositorio, a la derecha tenemos una serie de opciones, la primera es nuevo archivo.



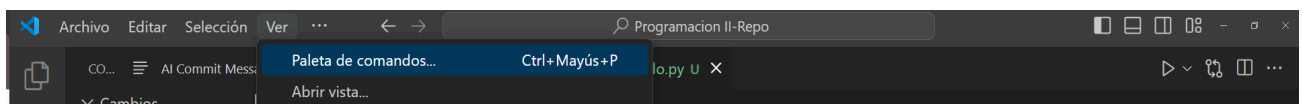
Hacemos clic y ya podemos definir el nuevo archivo. Escribimos el nombre del archivo, teniendo en cuenta que no solo debemos indicar el título, sino también su extensión. Por ejemplo, 'holamundo.py'. Incluimos el contenido del archivo en su interior y hacemos clic en Guardar. Si ahora nos dirigimos a la pestaña Cambios de Git, veremos que se ha producido un cambio en la carpeta que contiene el repositorio local. Y, ojo, esto es muy importante tenerlo en



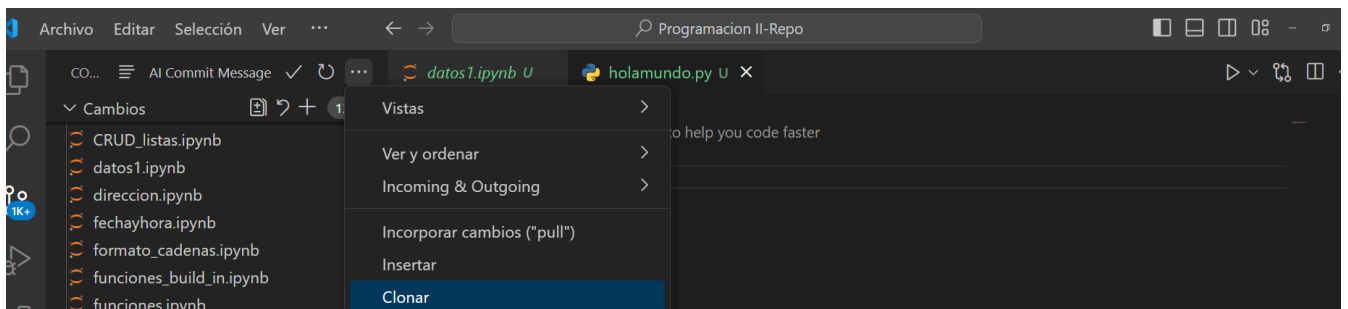
cuenta. Aunque hayamos creado un archivo nuevo en una carpeta que contiene un repositorio local, todavía no hemos sincronizado nuestro repositorio local con el repositorio remoto. De hecho, todavía ni siquiera hemos anotado el cambio en nuestro repositorio local, ya que aún no hemos hecho 'commit'. Por tanto, hemos creado un archivo en la carpeta que contiene el repositorio, pero aún no se ha anotado dicho cambio como tal en el repositorio. Por último, nuestro archivo puede ser editado como cualquier otro. Suponiendo que no tenemos el archivo abierto, simplemente haremos doble clic sobre el archivo, modificamos lo que necesitemos y le damos de nuevo al botón de guardar, teniendo claro una vez más que estamos guardando los cambios producidos en el archivo, aún no en el repositorio.

Conectarse con autenticación a un repositorio remoto a través de Visual Studio

Cuando empezamos a trabajar en un equipo probablemente existan repositorios ya creados, bien públicos o bien privados, a los que conectarnos a través de Visual Studio. Para conectarnos a repositorios remotos, ya sean públicos o privados, a través de Visual Studio nos dirigimos a la barra de menús, hacemos clic en Ver y a continuación Paleta de comandos...




También podríamos acceder a este mismo modal haciendo clic en Control de código fuente, y a la opción de los tres puntos horizontales para desplegar más opciones y elegir la opción Clonar.




Si el repositorio es público, entonces lo único que necesitaremos es copiar la URL del mismo y pegarla en este cuadro de texto. En cambio, si el repositorio es privado, solo podremos acceder a la URL si hemos iniciado sesión con nuestra cuenta de GitHub. El siguiente paso es localizar el repositorio que deseamos clonar. Podemos copiar la URL del repositorio o si ya tenemos repositorios en nuestra cuenta podemos seleccionarlos directamente





Proporcione la dirección URL del repositorio o seleccione un origen de repositorio.

 Clonar desde GitHub orígenes remotos

Nombre del repositorio (escribir para buscar)

 gpimentel79/Programacion-II-Repo <https://github.com/gpimentel79/Programacion-II-Repo.git>

Hacemos clic sobre el repositorio definido y elegimos dónde queremos clonar el repositorio.

 Clonando el repositorio GIT 


'https://github.com/gpimentel79/Trabajo-Final.git'...

Origen: GIT.

Cancelar

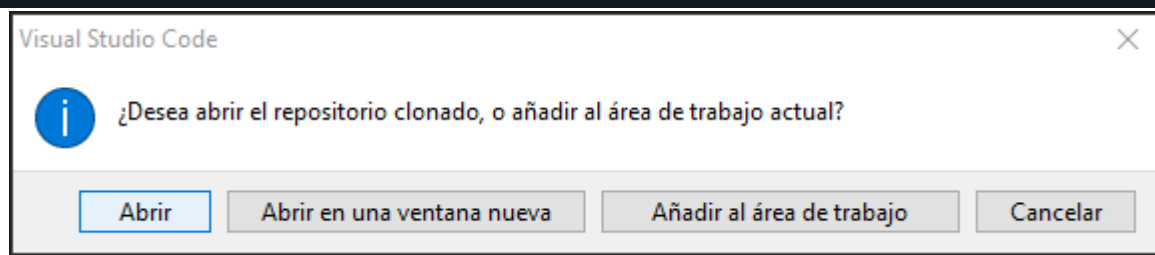
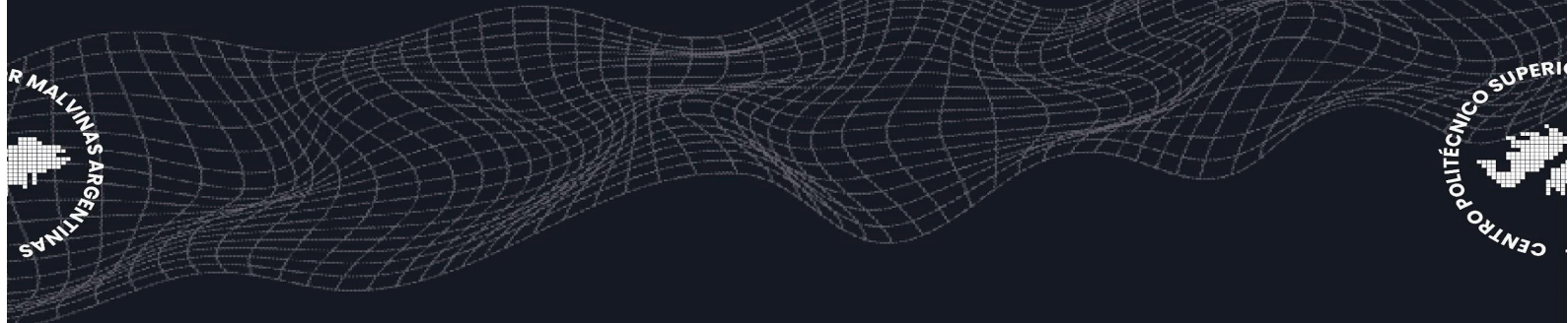
Una vez finalizado de clonar, nos preguntará si queremos abrir el repositorio clonado y añadirlo al área de trabajo.

Visual Studio Code

 ¿Desea abrir el repositorio clonado, o añadir al área de trabajo actual?

Abrir **Abrir en una ventana nueva** **Añadir al área de trabajo** **Cancelar**

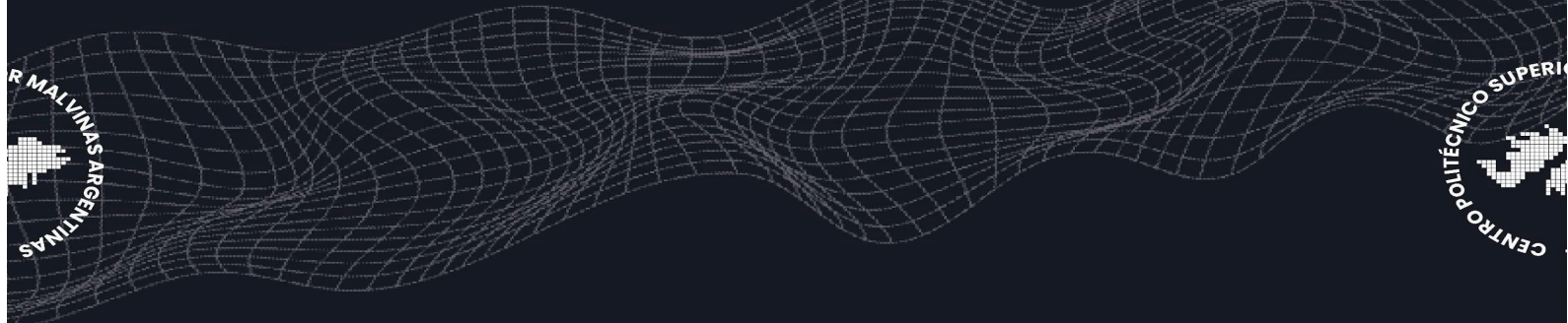
Cuando empezamos a trabajar en un equipo probablemente existan repositorios ya creados a los que conectarnos a través de Visual Studio.



Bajar cambios del repositorio remoto en GitHub al repositorio local mediante Visual Studio

Cuando empezamos nuestra jornada laboral debemos alinearlos con el trabajo realizado por nuestro equipo y que ya se encuentra en el repositorio remoto, vamos a ver como bajarnos los cambios del repositorio remoto alojado en GitHub a nuestro proyecto local, haciendo uso de Visual Studio. Nuestro proyecto en Visual Studio presenta una serie de cambios que todavía no han sido ni siquiera anotados en el repositorio local en cuestión. Sin embargo, antes de hacer ningún cambio, es una buena práctica traernos desde el repositorio remoto a nuestro repositorio local todos los cambios que se han producido desde la última sincronización, ya que durante este tiempo puede ser que algún colaborador o colaboradora del proyecto haya realizado algún cambio y lo haya subido al repositorio remoto. Para traer a nuestro repositorio local aquellos cambios que se hubieran producido en el repositorio remoto, nos dirigimos a la opción Control de código fuente y a continuación localizamos la el icono con tres puntos horizontales de Más acciones que se encuentra en la parte superior y hacemos clic “Pull, Push”, y luego la opción Extraer de... Si nos dirigimos ahora al explorador de soluciones, como podemos ver, nuestro repositorio local se ha actualizado.





Cuando empezamos nuestra jornada laboral debemos alinearlos con el trabajo realizado por nuestro equipo y que ya se encuentra en el repositorio remoto.

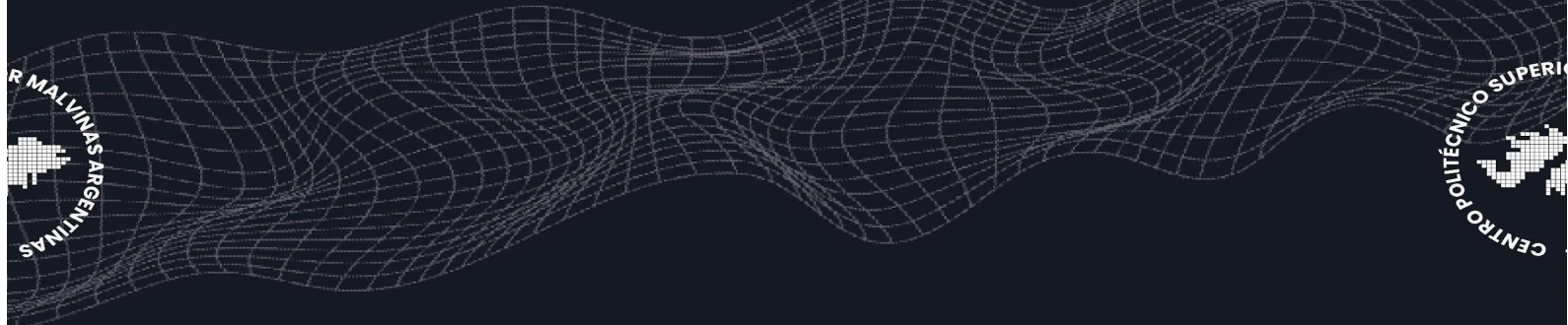
Subiendo cambios del repositorio local al repositorio remoto a través de Visual Studio

A medida que vamos terminando de implementar funcionalidades deberíamos ir subiéndolas al repositorio remoto alojado en GitHub para que otros compañeros puedan continuar trabajando sobre nuestro progreso. Supongamos que ya hemos realizado una serie de cambios en nuestro proyecto y que hemos hecho un 'commit' de dichas modificaciones. Por tanto, nuestro repositorio local ya puede enviar el historial de cambios de nuestro proyecto al repositorio remoto. En la jerga del control de versiones es lo que se conoce como hacer 'push' o subir cambios al repositorio remoto. Para hacer 'push', nos dirigimos a la opción Control de código fuente y a continuación localizamos la el icono con tres puntos horizontales de Más acciones que se encuentra en la parte superior y hacemos clic “Pull, Push”, y luego la opción Insertar en...

Seleccionar un elemento remoto para publicar la rama "master":

origin <https://github.com/gpimentel79/Programacion-II-Repo.git>

Los cambios anotados en nuestro proyecto local ya están también disponibles en el repositorio remoto. Y si accedemos al mismo y recargamos la página, veremos que efectivamente nuestros dos repositorios, el local y el remoto, ya son iguales.



Programacion-II-Repo / clase2 / `holamundo.py`



gpimentel79 agregando mensaje de cambio en repositorio remoto

23baa60 · 7 minutes ago

History

Code

Blame

2 lines (2 loc) · 82 Bytes



Code 55% faster with GitHub Copilot

Raw



```
1 print("hola mundo!")
2 print("haciendo un commit e impactando el cambio en remoto")
```

La rama "master" en Git

Para comenzar con la gestión de ramas, debemos partir desde la rama principal, también conocida como rama "máster". Hasta el momento hemos estado trabajando con transferencias de archivos muy sencillas entre nuestro repositorio local y nuestro repositorio remoto mediante acciones como 'commit' y 'push' para enviar nuestra información al repositorio remoto y extraer para traernos la información a nuestro repositorio local. Todas estas acciones las hemos realizado sobre la rama máster, que no es más que la rama por defecto de Git. Máster ha sido históricamente su nombre, pero podríamos ponerle el nombre que queramos accediendo a la configuración del repositorio. Una rama representa una línea independiente de desarrollo. Podemos pensar en una rama como una forma de solicitar un nuevo directorio de trabajo, un nuevo entorno de pruebas o una nueva versión del proyecto. En GitHub podremos ver en qué rama nos encontramos haciendo clic sobre este desplegable.



master 1 Branch 0 Tags

Go to file t Add file <> Code

About

Switch branches/tags

Find or create a branch...

Branches Tags

✓ master default

View all branches

actividad10.ipynb

o en repositorio remoto 23baa60 · 2 days ago 15 Commits

agregando mensaje de cambio en repositorio remoto	2 days ago
se agregó una variable	8 months ago
Mi primera versión de archivo de variables	8 months ago
Mi primera versión de archivo de variables	8 months ago
se agregó una variable	8 months ago

No description, website, or topics provided.

Activity

0 stars

1 watching

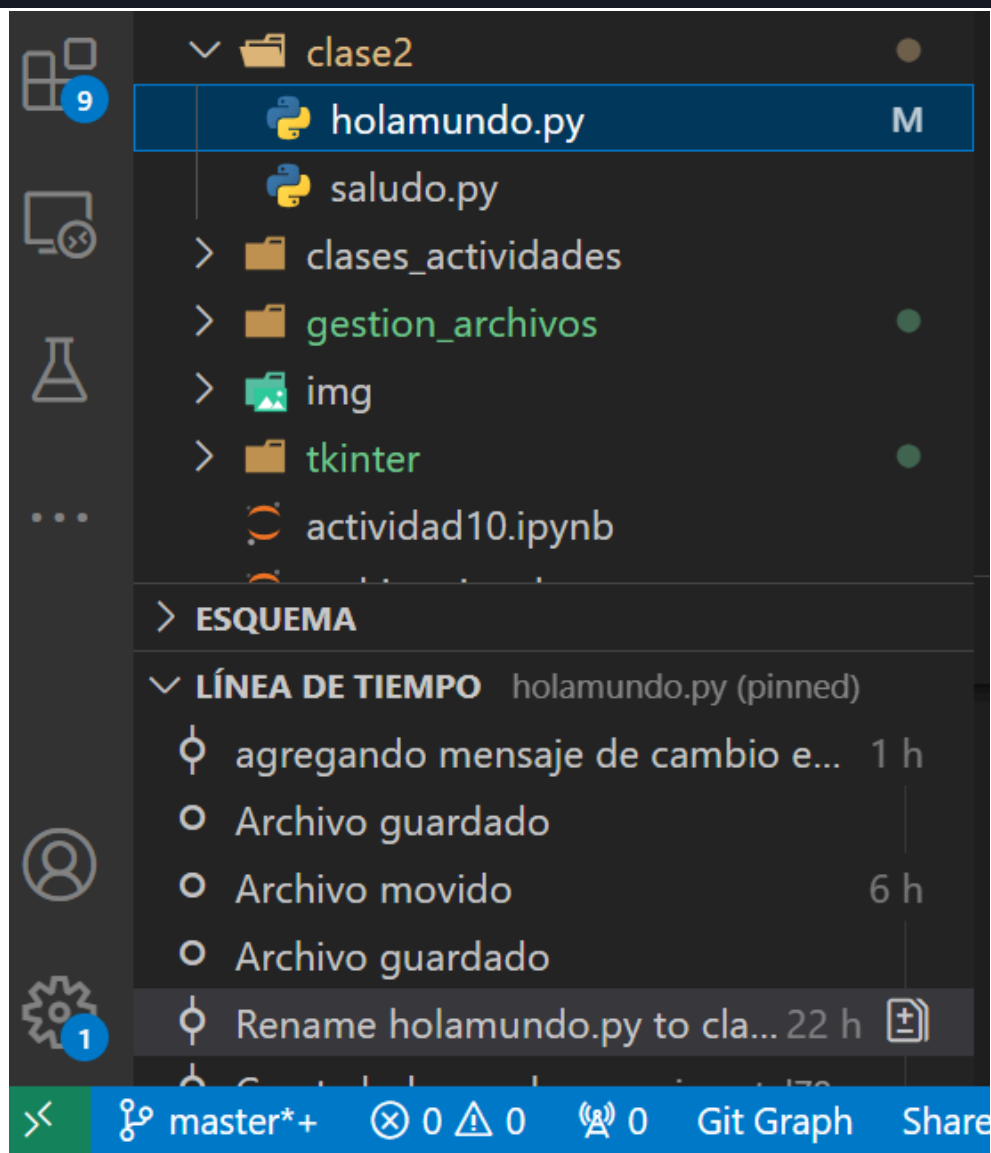
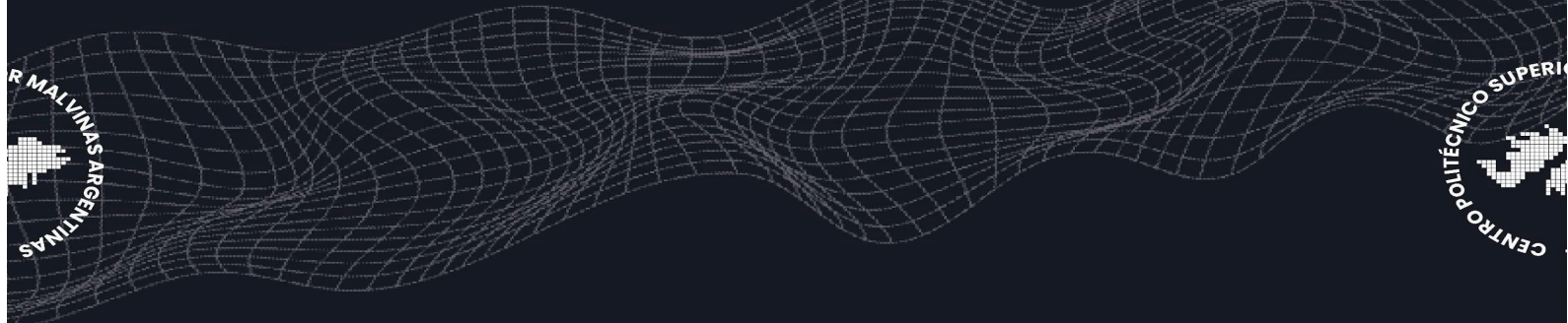
0 forks

Releases

No releases published

Create a new release

Como podemos ver, la rama máster es la que se encuentra por defecto. Y en Visual Studio podemos también ver en qué rama estamos trabajando en la zona inferior



La rama máster comúnmente es la rama de referencia de nuestro proyecto, así que debemos tener mucho cuidado con el código que subimos ahí. La rama máster debe contener la versión más estable y que mejor funciona de nuestro proyecto. Para comenzar con la gestión de ramas debemos partir de la rama principal, también conocida como rama máster.

Introducción a Git Flow para la organización de ramas

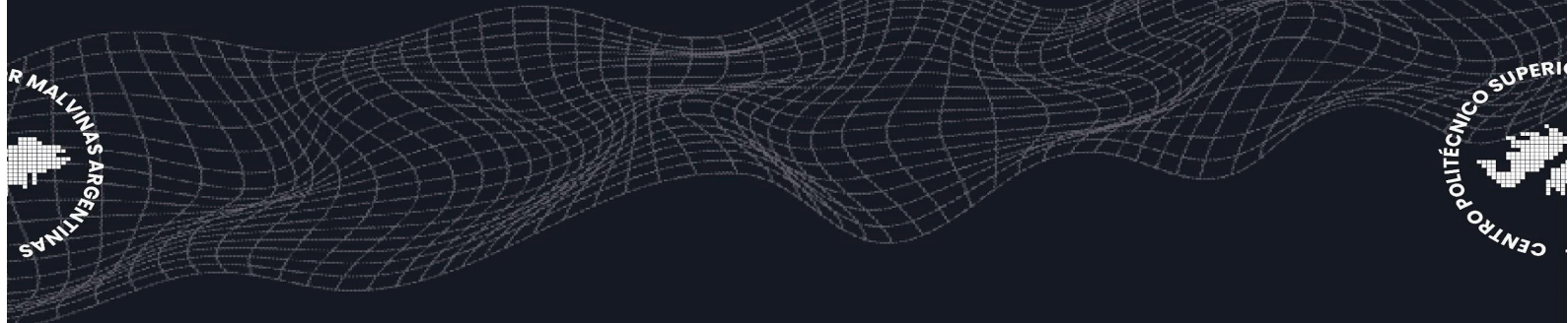
Cuando trabajamos en equipo, la gestión de ramas puede volverse un proceso



complicado. Entonces, ¿cómo organizamos las ramas en nuestro repositorio? Git Flow es una estrategia presentada por un ingeniero de software holandés allá por el año 2010 con el enfoque de mejorar la organización de ramas dentro de nuestros proyectos, aportando más fluidez en el día a día. Git Flow se adapta muy bien a la colaboración y al crecimiento tanto del proyecto como del equipo, ya que permite que muchas personas, tantas como sean necesarias, puedan trabajar en paralelo de un modo organizado. Git Flow nos plantea dos ramas principales, máster y develop, siendo develop una rama que sale de la rama máster. Cuando empezamos a trabajar en una nueva funcionalidad o conjunto de funcionalidades no vamos a volcar código directamente sobre máster, sino que deberíamos saltar a la rama develop. Pero resulta que en la rama develop tampoco deberíamos volcar código directamente, sino que esta funcionalidad o conjunto de tareas que vamos a implementar va a dividirse en nuevas ramas, probablemente una rama por funcionalidad o por tarea, de modo que los diferentes miembros del equipo puedan trabajar en paralelo.



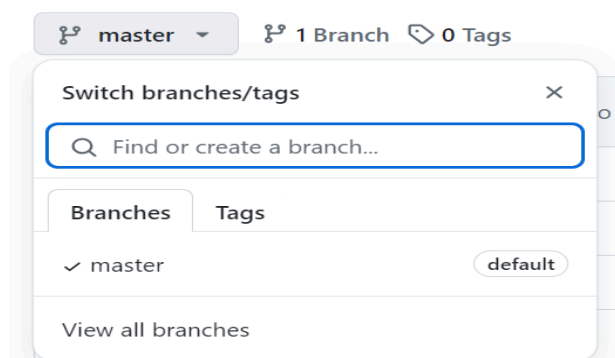
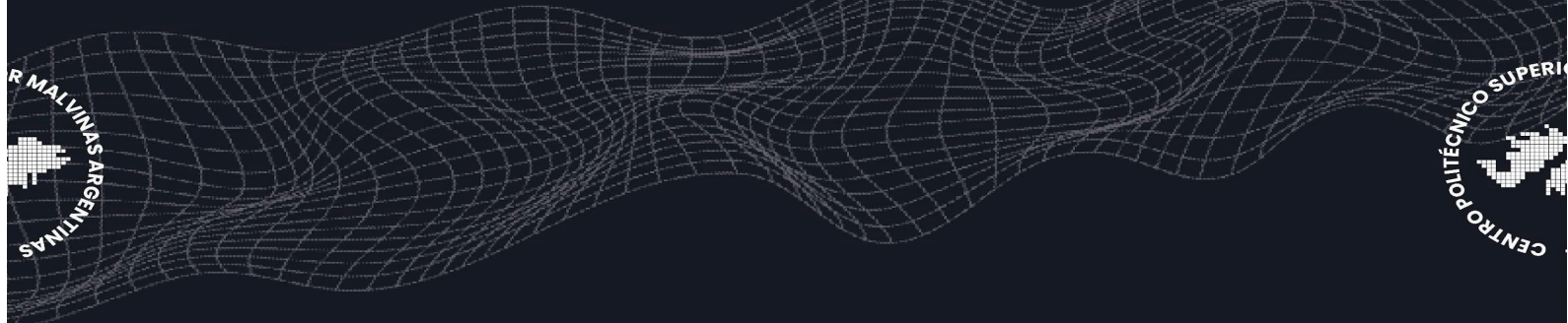
En el momento en el que un desarrollador o desarrolladora termine su tarea



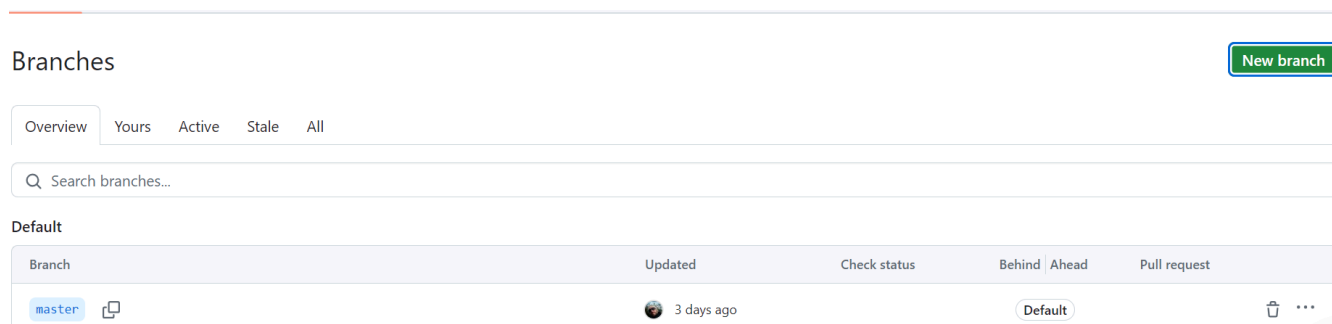
volcará su código desde la rama para dicha tarea hasta la rama develop. Es importante entender que el código va de una rama a otra. Por puras cuestiones de seguridad, la rama máster y la rama develop nunca deberían recibir anotaciones directamente, sino siempre a través de otra rama. De modo que máster únicamente recibiría código que ya está en la rama develop, y develop solo recibiría código desde ramas secundarias creadas para la implementación de diferentes funcionalidades. Puede ocurrir que se entreguen funcionalidades concretas a la rama máster mientras que otros miembros del equipo continúan desarrollando código en sus ramas secundarias. Sea como sea, siempre trataremos de mantener la estrategia de ramificación propuesta en Git Flow, ya que entre rama y rama es posible establecer diferentes mecanismos de revisión de código, comprobación de la calidad del mismo mediante cobertura de test y otra serie de pasos que podríamos incluir para evitar que código erróneo termine afectando a usuarios finales que utilicen nuestro software.

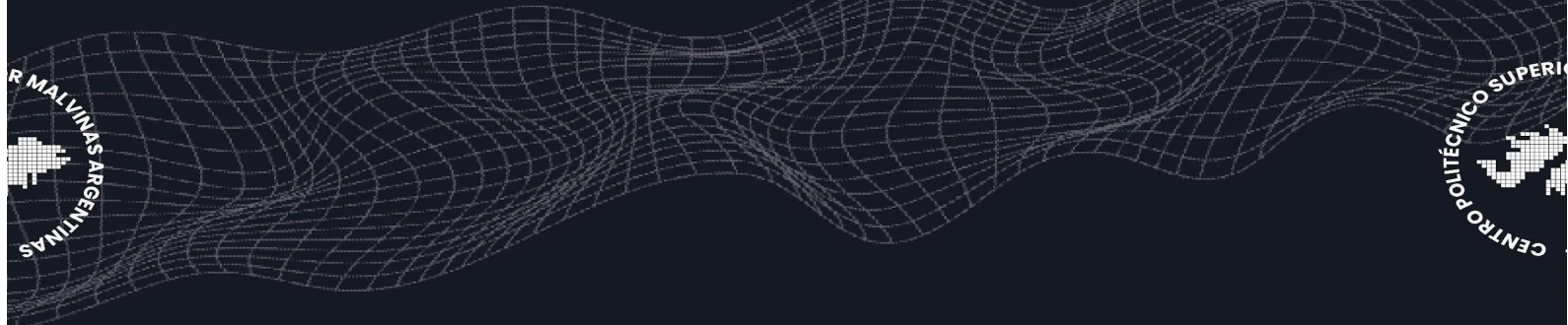
Creación de una rama en GitHub

GitHub nos permite crear ramas a partir de un repositorio remoto alojado en la plataforma. Para crear una rama desde la interfaz de GitHub, accedemos a la URL del repositorio en cuestión y, habiendo iniciado sesión, localizamos este recuadro que nos indica la rama en la que nos encontramos actualmente. Abrimos el desplegable y visualizamos todas las ramas haciendo clic en View all branches (ver todas las ramas).



También podríamos acceder a esta vista desde la página anterior haciendo clic en este enlace [🔗 1 Branch](#) que se encuentra a la derecha del recuadro en cuestión. Desplazándonos hacia esta zona de la pantalla, encontramos un icono con el texto New branch (nueva rama), hacemos clic y a continuación escribimos en el modal que nos aparece el nombre de la rama que deseamos crear. Por ejemplo, 'miPrimeraRama'. Es muy importante elegir correctamente cuál será la rama de origen, ya que desde la rama origen abriremos una nueva bifurcación, que será esta rama que estamos creando. En estos momentos vamos a crear una nueva rama llamada 'miPrimeraRama' a partir de la rama máster y hacemos clic en Create new branch (crear nueva rama). Y, listo, ya hemos creado una nueva rama en nuestro repositorio remoto alojado en GitHub.





Create a branch

New branch name

miPrimeraRama

Source

master

Cancel

Create new branch

Branches

[New branch](#)

Overview **Yours** Active Stale All

Q Search branches...

Default

Branch	Updated	Check status	Behind	Ahead	Pull request
master	3 days ago				Default

Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
miPrimeraRama	11 minutes ago		0	0	

Active branches

Branch	Updated	Check status	Behind	Ahead	Pull request
miPrimeraRama	11 minutes ago		0	0	

Actividad:

Como actividad de la clase vamos a practicar con Github y VsCode lo que vimos en la clase

Instancia de Autoevaluación:



En esta instancia de autoevaluación te pedimos que puedas poner en juego lo que has aprendido hasta este momento. Recordá que siempre podrás volver a la teoría presentada en esta clase o hacer las preguntas pertinentes en los encuentros sincrónicos o las tutorías presenciales.

Es importante tener en cuenta que para aprobar esta instancia deberás:

- Entregar en tiempo y forma
- Realizar un 60% de la actividad correctamente.
- Cumplir con las consignas
- Cumplir con el formato requerido.

Cierre:

Llegamos al final de la tercera clase 🤖. En esta clase hemos explorado temas fundamentales como la gestión de archivos en GitHub y Visual Studio, la colaboración y sincronización de repositorios, así como la creación y eliminación de ramas. Estos conocimientos son esenciales para desarrollar software de manera eficiente y colaborativa.

Recuerden que la práctica es clave para afianzar lo aprendido. Los invito a seguir explorando GitHub y Visual Studio Code, y a poner en práctica sus habilidades en el control de versiones y la colaboración en proyectos de software.

En la próxima clase, finalizamos el control de versiones para trabajar en equipos de desarrollo y vamos a ver herramientas para potenciar nuestro trabajo gracias a la Inteligencia Artificial. ¡Prepárense para seguir aprendiendo y desarrollando sus habilidades en programación! 🚀🏠



Bibliografía Obligatoria:

- Ramírez Jiménez, Ó. (2021). *Python a fondo*. Marcombo.

Recursos adicionales:

- ¿Qué es un Jupyter Notebook? (2021, October 4). YouTube. Retrieved April 7, 2023, from <https://www.youtube.com/watch?v=G4K1jKZH7nE>
- Entrenamiento de programación en Python 3 - Nivel básico. (2023, January 1). Programación en Python. Retrieved April 7, 2023, from <https://entrenamiento-python-basico.readthedocs.io/es/3.7/>
- (n.d.). El Libro De Python: 🏠 Inicio. Retrieved April 7, 2023, from <https://ellibrodepython.com/>
- Curso Python. Vídeo 1. (2017, January 24). YouTube. Retrieved April 7, 2023, from <https://www.youtube.com/watch?v=G2FCfQj-9ig>
- (n.d.). Visualize Code Line-by-Line. Retrieved April 7, 2023, from <https://memlayout.com/>
- Python in Visual Studio Code. (n.d.). Visual Studio Code. Retrieved April 7, 2023, from <https://code.visualstudio.com/docs/languages/python>