

TUTORIAL GITHUB

Empecemos con algo sencillo, ¿Qué es GitHub? Github es una plataforma en línea la cual funciona como un repositorio gratuito en el cual se permite manipular y gestionar proyectos y sus versiones.

En este tutorial, se explicará la manejabilidad y el uso de GitHub. Dentro de este documento se explica cómo crear una cuenta de GitHub, crear “commits” de código, manejar las diferentes versiones de un repositorio y cómo compartir el código con otros colaboradores. Además de esto, se agregaron tips como lo pueden ser el cómo crear una carpeta y ver el historial de versiones.

¿CÓMO ACCEDER AL TUTORIAL?

Para acceder a toda la información necesaria para el tutorial de GitHub y de Prolog, da click en el siguiente link: ([link](#)). Este link es un repositorio donde se encuentran guardados dichos tutoriales.

GLOSARIO

Repository: Lugar donde se almacena y se realiza la distribución del código de una aplicación o un programa.

Push Request: Incorporación de cambios, son modificaciones o cambios los cuales forzados a estar presentes en el repositorio escogido.

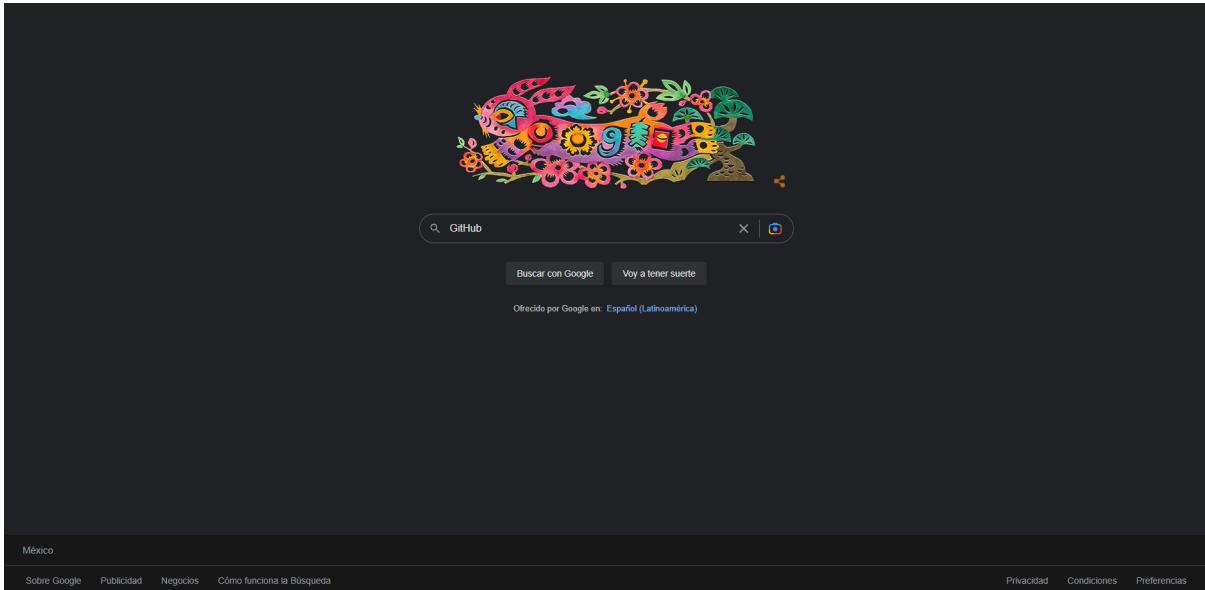
Pull Request: Solicitud de incorporación de cambios, son modificaciones propuestas para un repositorio enviadas por un usuario y que los colaboradores del repositorio aceptan o rechazan.

Commit: Cambio individual a un archivo, o conjunto de archivos. También conocido como revisión,

Merge: La fusión, o merge, toma los cambios de una rama (dentro del mismo repositorio) y se aplican en otra rama.

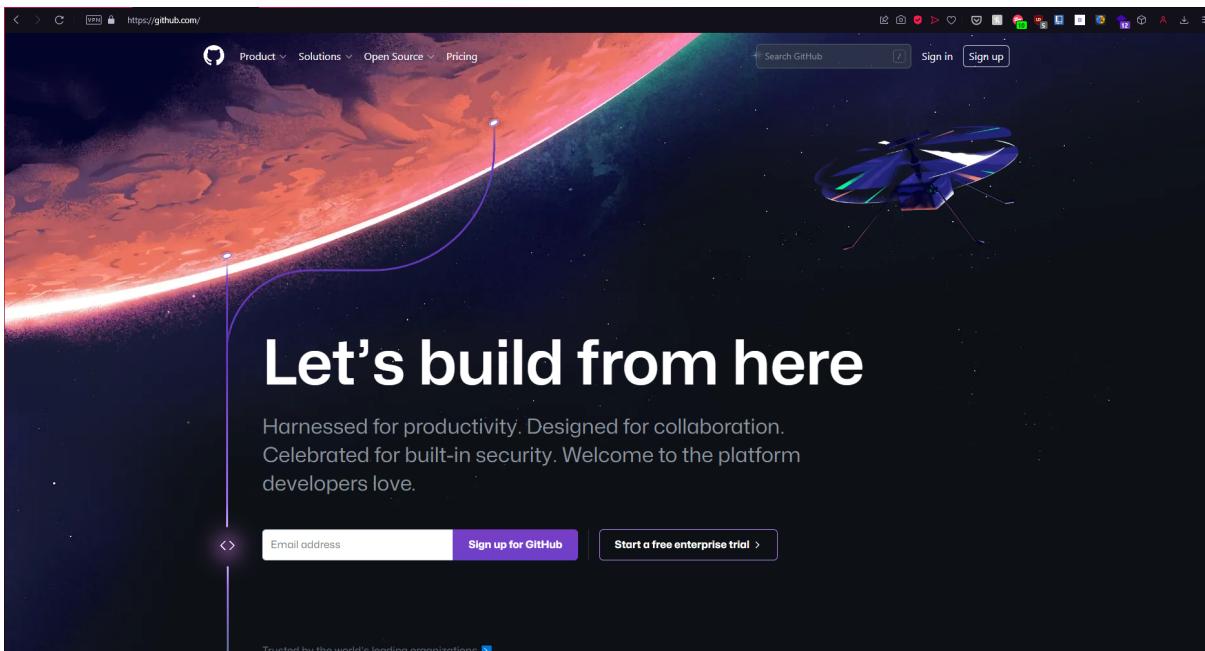
¿CÓMO CREAR UNA CUENTA DE GITHUB?

PRIMER PASO



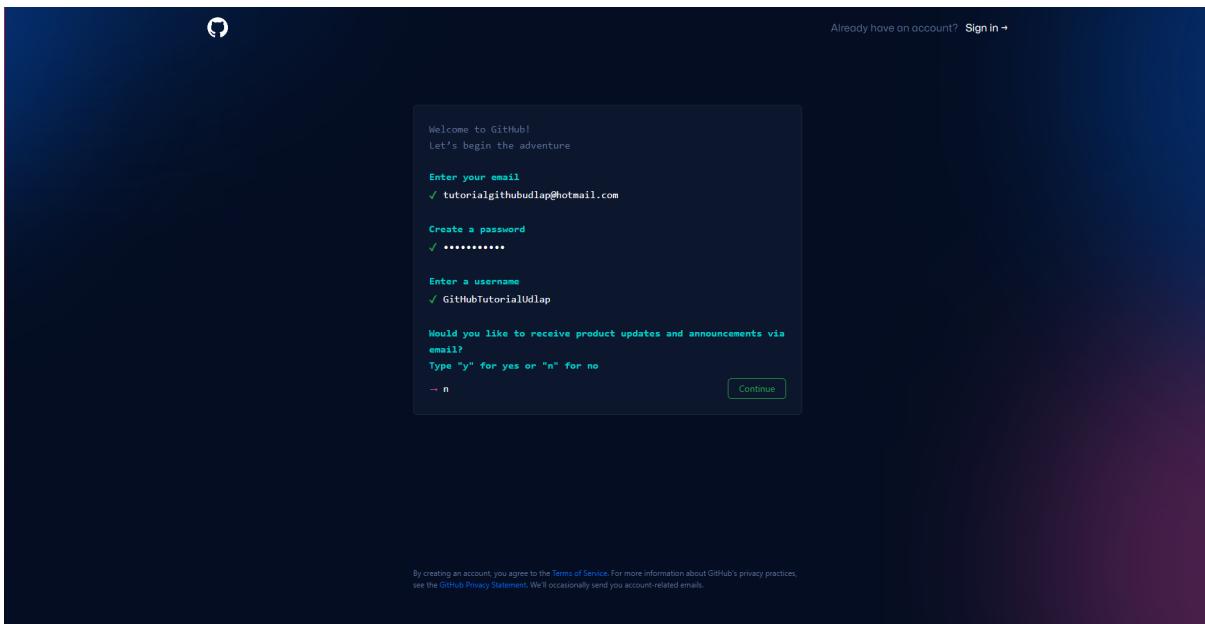
Abre tu navegador de confianza e ingresa a cualquier motor de búsqueda.

SEGUNDO PASO



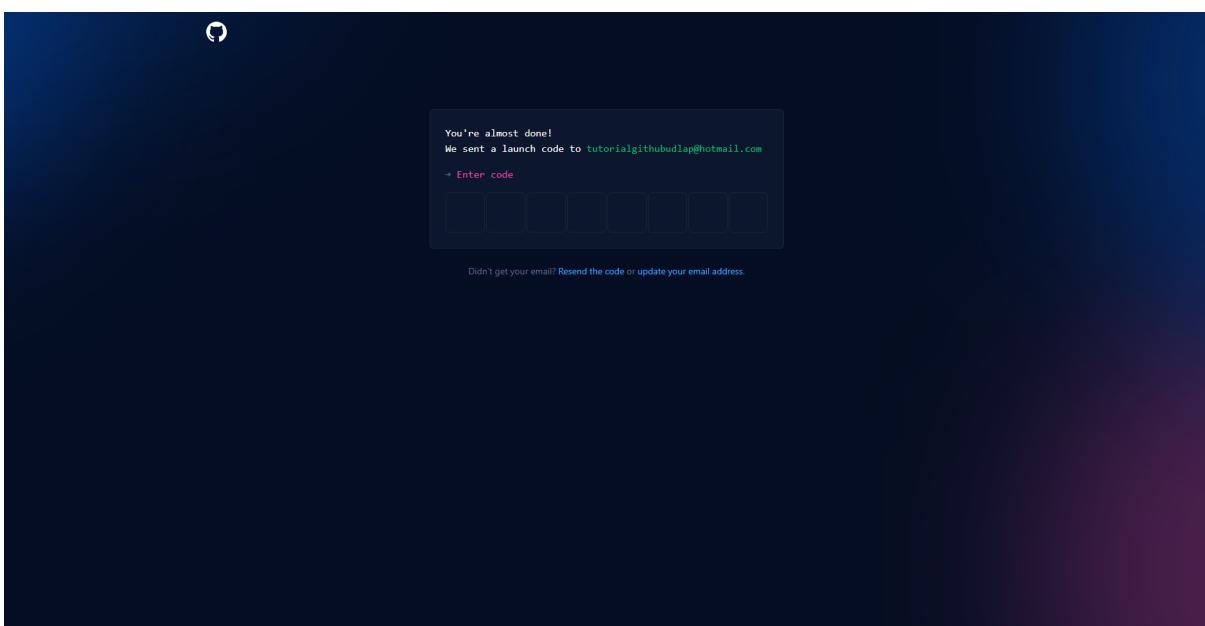
Navega a <https://github.com/> y haz clic izquierdo en el botón que dice "Registrarse" en la esquina superior derecha.

TERCER PASO



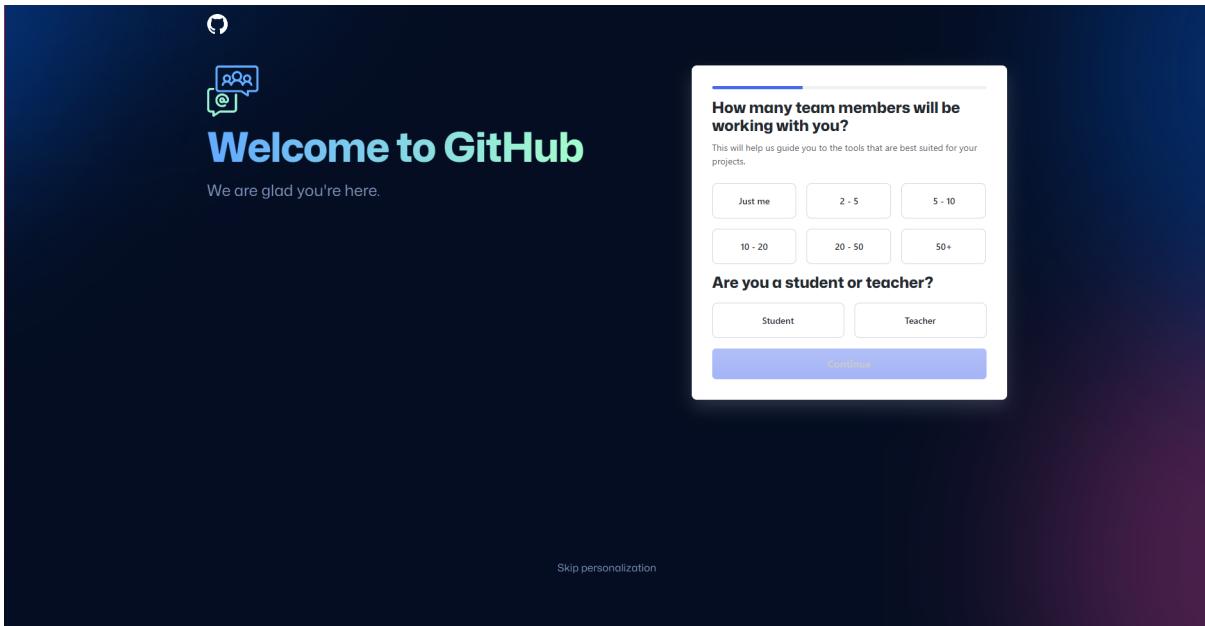
Ingresa tu dirección de correo electrónico y una contraseña segura en los campos correspondientes. Haz clic en el botón "Registrarse".

CUARTO PASO



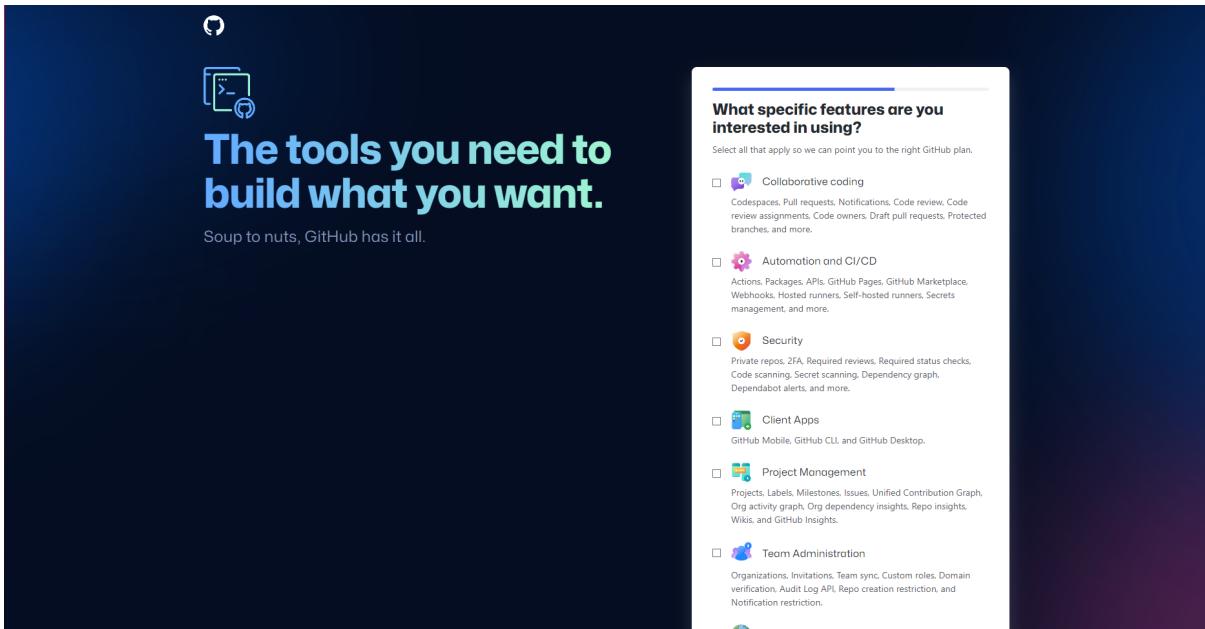
Verifica tu dirección de correo electrónico ingresando el código de verificación que se te ha enviado.

QUINTO PASO



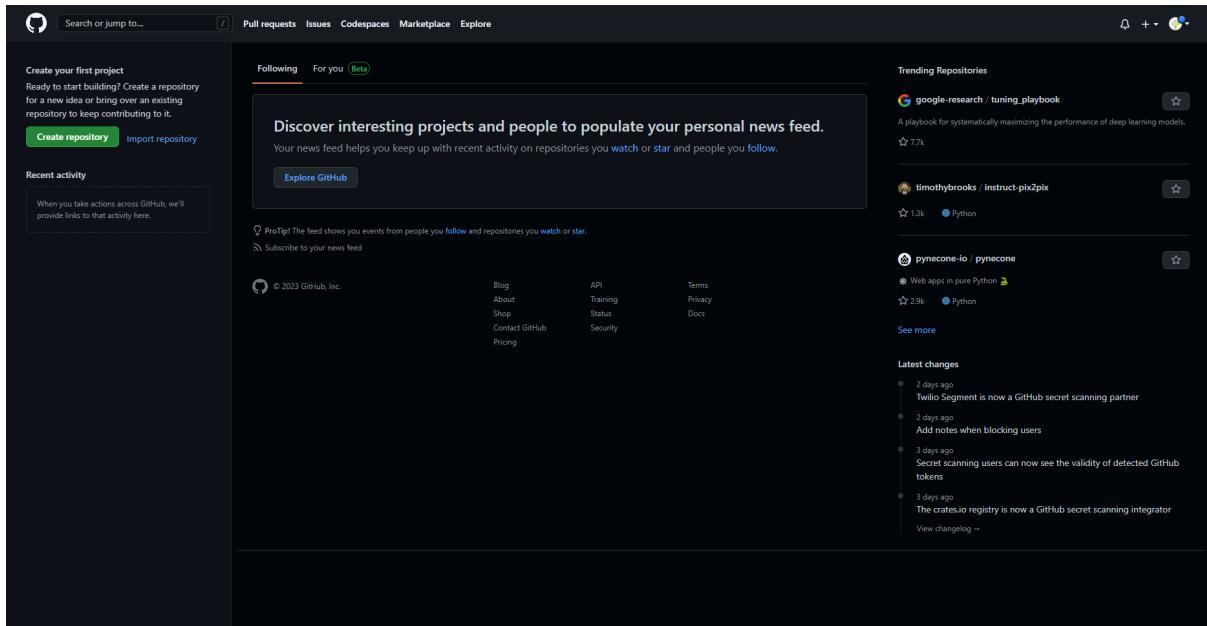
Escoja el número de personas con las que planea trabajar y especifique si es estudiante o maestro.

SEXTO PASO (OPCIONAL)



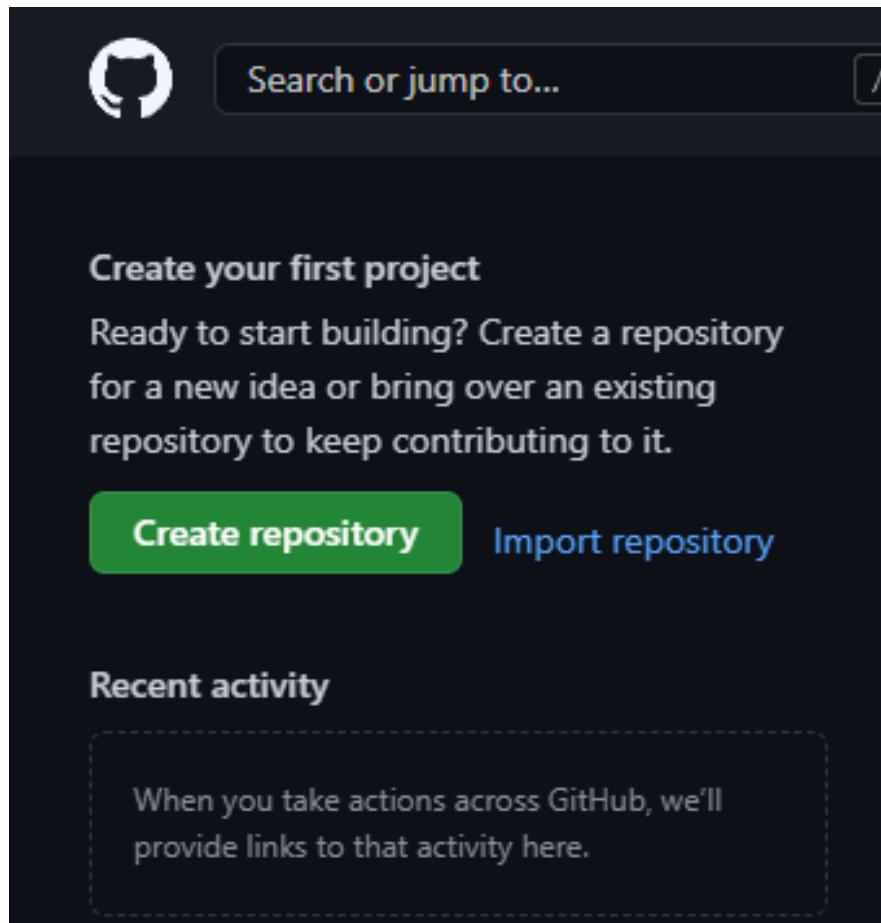
Personaliza tu GitHub con las diferentes funciones que la plataforma maneja. No es obligatorio seleccionar algo.

SÉPTIMO PASO

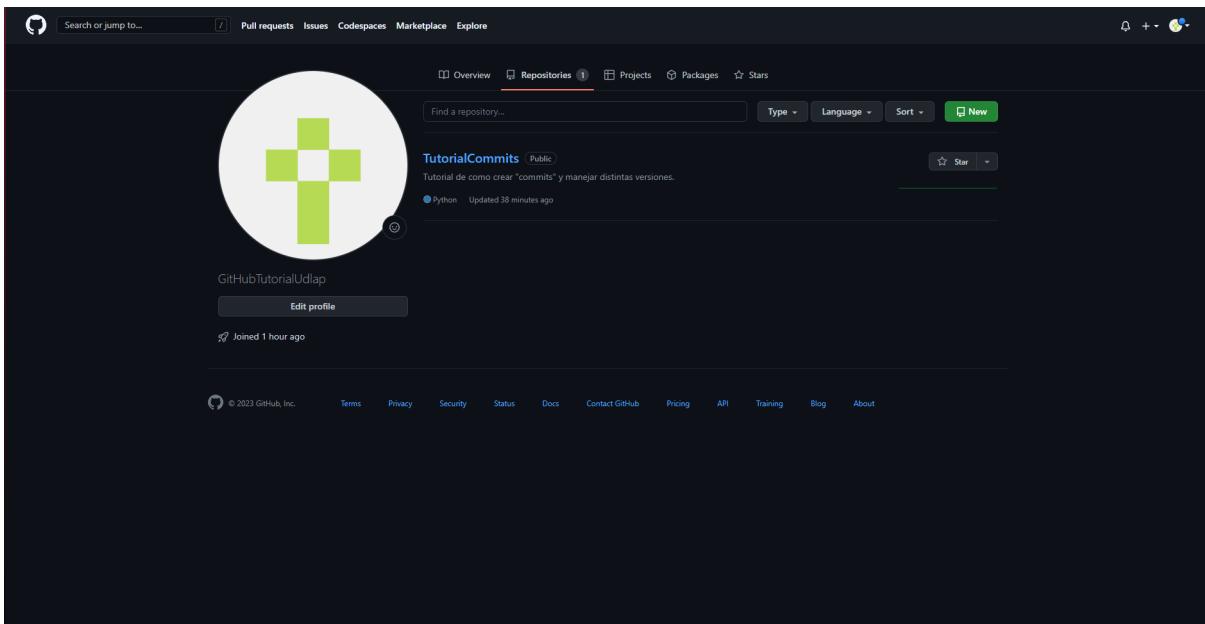


Completa tu perfil, agregando una imagen de perfil y una descripción si lo deseas.

OCTAVO PASO



En caso de no tener ningún repositorio creado, selecciona el botón { Create repository }.

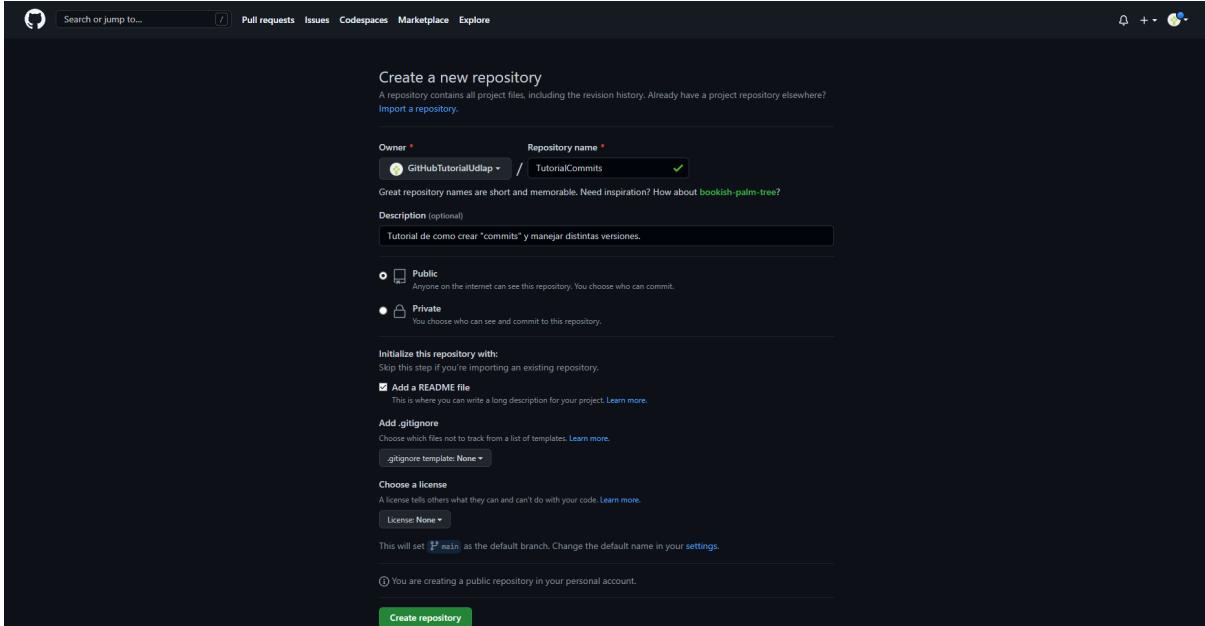


En caso de que ya hayas creado algún repositorio, debes hacer clic en el botón { New } que se encuentra en la parte superior derecha.

¡Enhorabuena! Ya tienes una cuenta de GitHub y ya sabes como crear un repositorio.

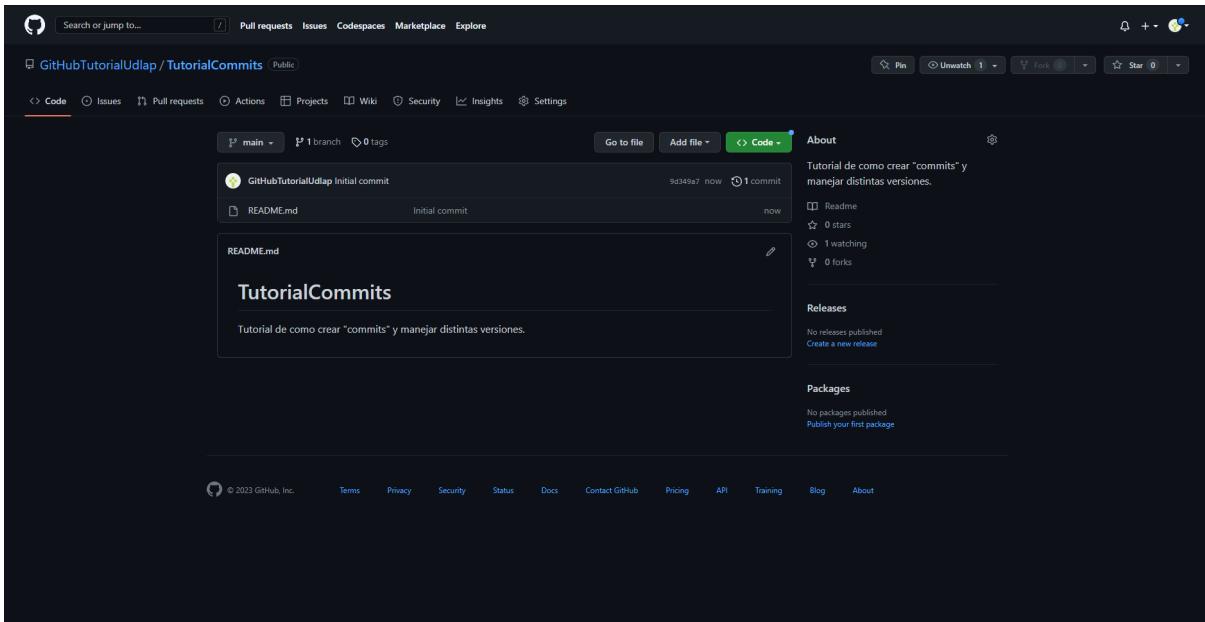
CREAR “COMITS” DE CÓDIGO

PRIMER PASO



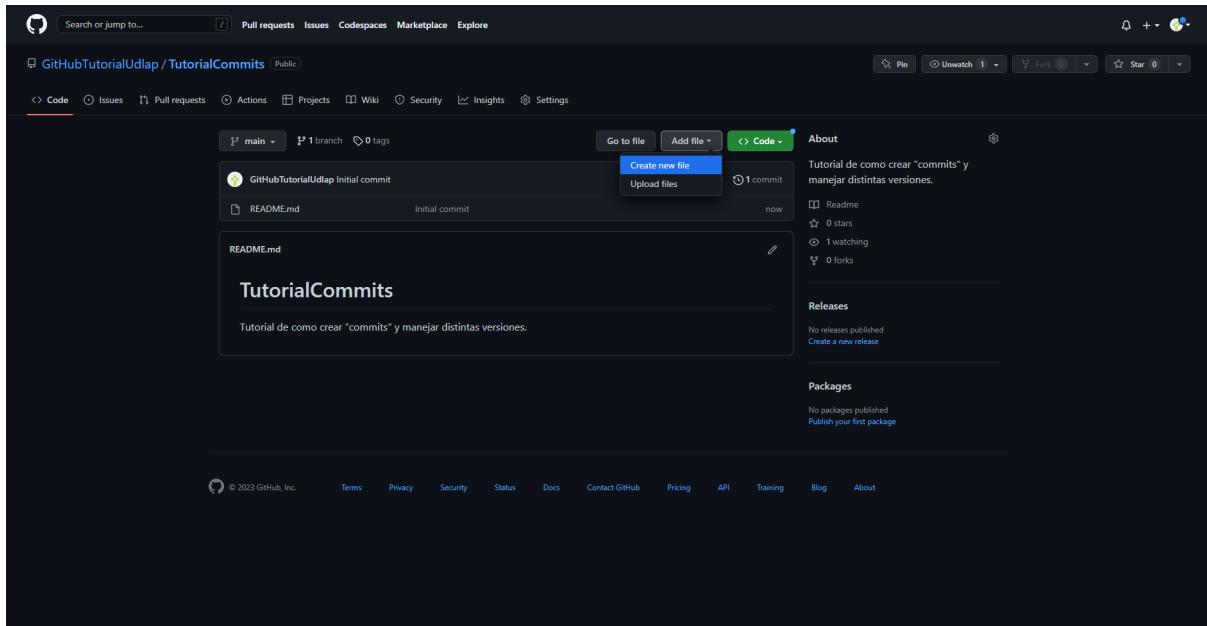
Crea un nuevo repositorio o selecciona uno existente en el que deseas hacer un "commit", también selecciona la opción "Add a README file".

SEGUNDO PASO



Así es como se ve el repositorio una vez se haya creado. Dentro de este repositorio, se puede escribir código, subir archivos existentes, buscar archivos dentro del repositorio y ver los cambios que se han hecho del mismo repositorio.

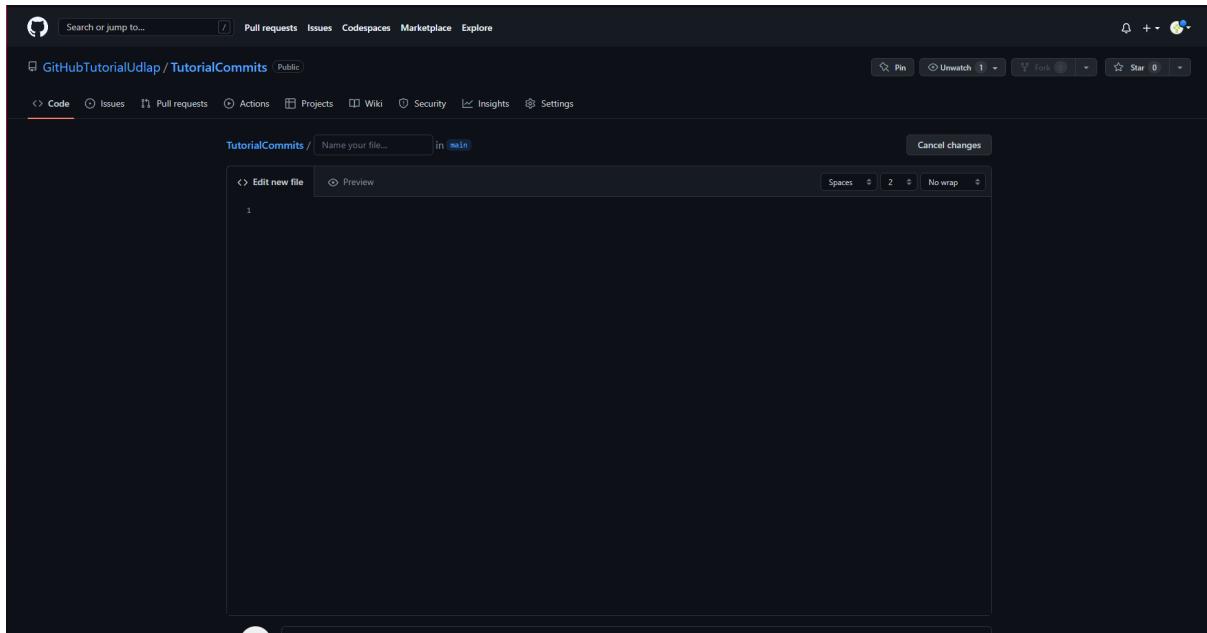
TERCER PASO

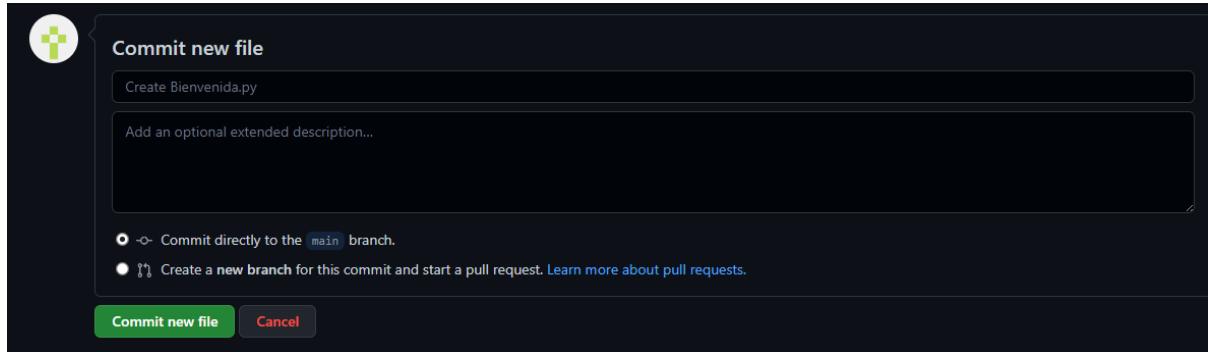


Haz clic en el botón "Create new file" en la página del repositorio para crear un nuevo archivo. Ese botón se encuentra en el apartado de "Add file".

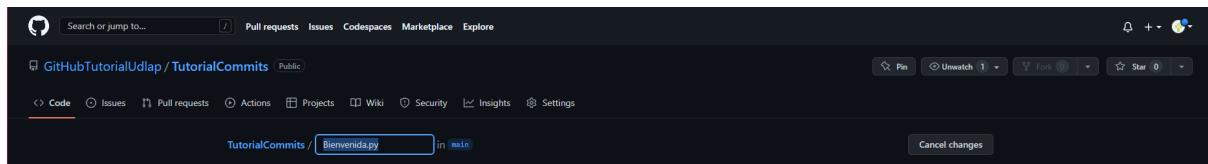
CUARTO PASO

La siguiente página aparece en la cual se puede modificar el archivo por completo al igual dónde se pueden hacer commits.



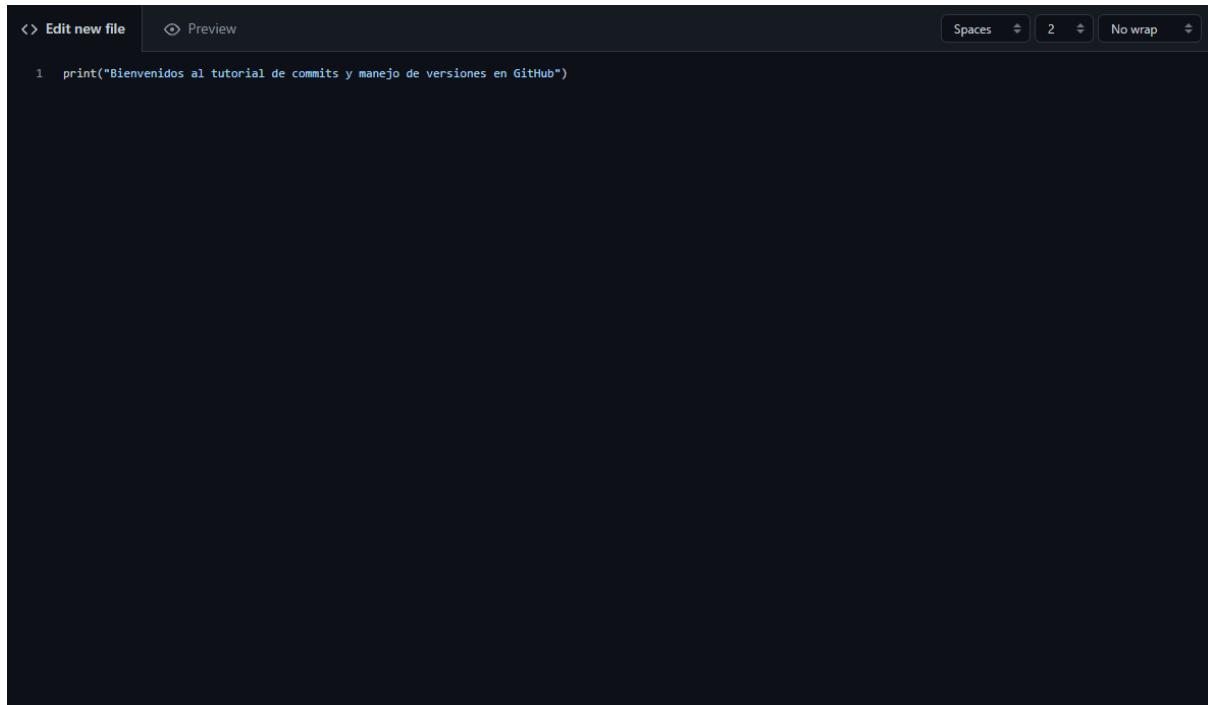


QUINTO PASO



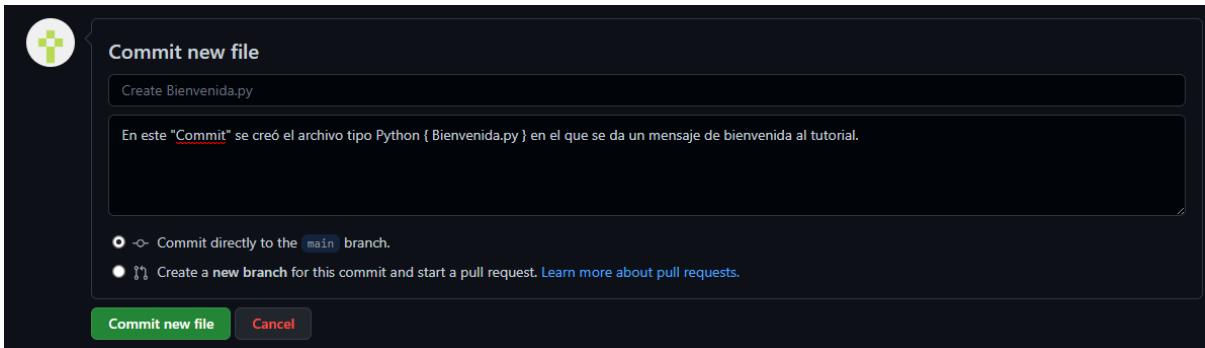
Nombrar el archivo recién creado.

SEXTO PASO



Agregar el código al archivo. Es importante mencionar que no solo se puede agregar código si no que cualquier tipo de archivo.

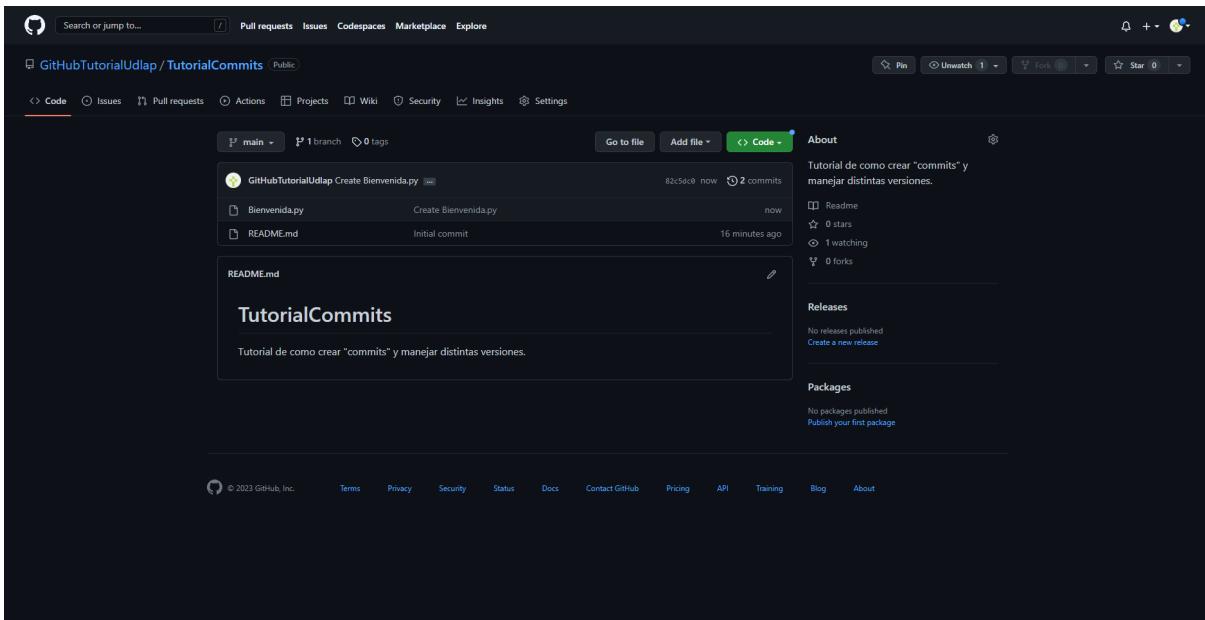
SÉPTIMO PASO



Una vez completo el código o el archivo que se creó, se selecciona si este archivo se mantendrá en la versión principal del repositorio o si se creará una versión nueva o “new branch” con este archivo. Para finalizar, seleccione el botón “Commit new file”.

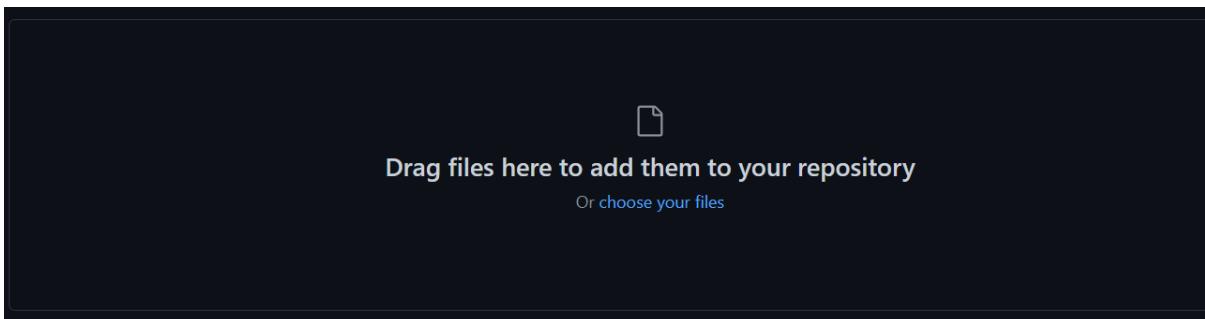
*ES MUY RECOMENDABLE, y una buena práctica, que se agregue una descripción del mismo archivo para tener un mejor control de versiones.

OCTAVO PASO



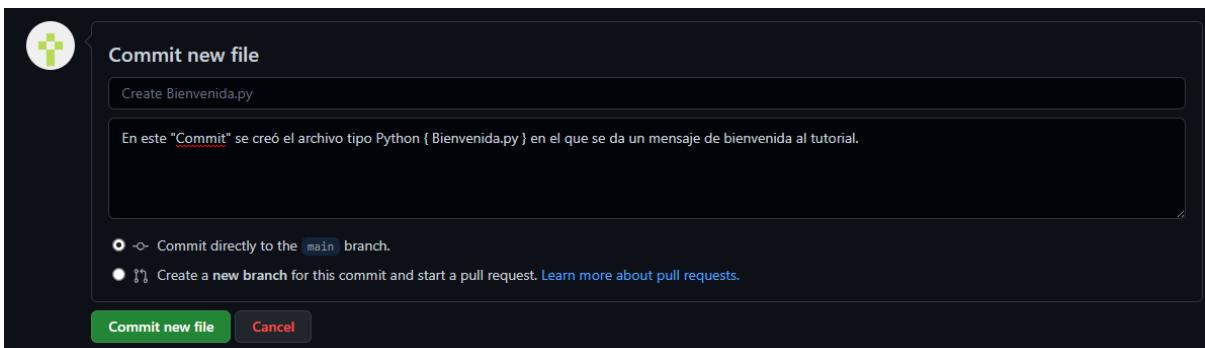
Vista del página principal del repositorio donde se pueden observar y acceder a los diferentes archivos dentro del repositorio. Haz clic en el botón "Upload files" en la página del repositorio para subir un archivo existente. Ese botón se encuentra en el apartado de "Add file".

NOVENO PASO



Arrastra o selecciona el apartado de “choose your files” y escoge el archivo o archivos existentes que quieras usar. Se puede seleccionar una carpeta completa.

DÉCIMO PASO



Una vez completo el código o el archivo que se creó, se selecciona si este archivo se mantendrá en la versión principal del repositorio o si se creará una versión nueva o “new branch” con este archivo. Para finalizar, seleccione el botón “Commit new file”.

PARA MOVERSE ENTRE CARPETAS



Ejemplo de cómo se ve una carpeta dentro de otra.

PRIMER PASO

Para agregar una carpeta, se debe entrar a “Add new file”, poner el nombre deseado del folder seguido de { / } la barra en diagonal. Eso creará una carpeta.

SEGUNDO PASO

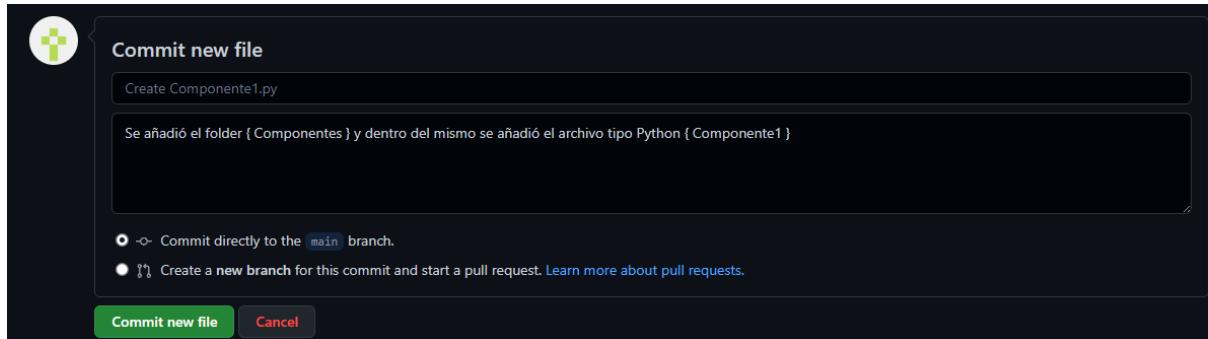
The screenshot shows a GitHub code editor interface. The top bar displays the repository path "TutorialCommits / Componentes / Componente1.py" and the branch "main". On the right, there are buttons for "Cancel changes", "Spaces", "2", and "No wrap". Below the header, there are two tabs: "Edit file" and "Preview changes". The main area contains the Python code:

```
1 print("Este es el primer componente del código")
2 |
```

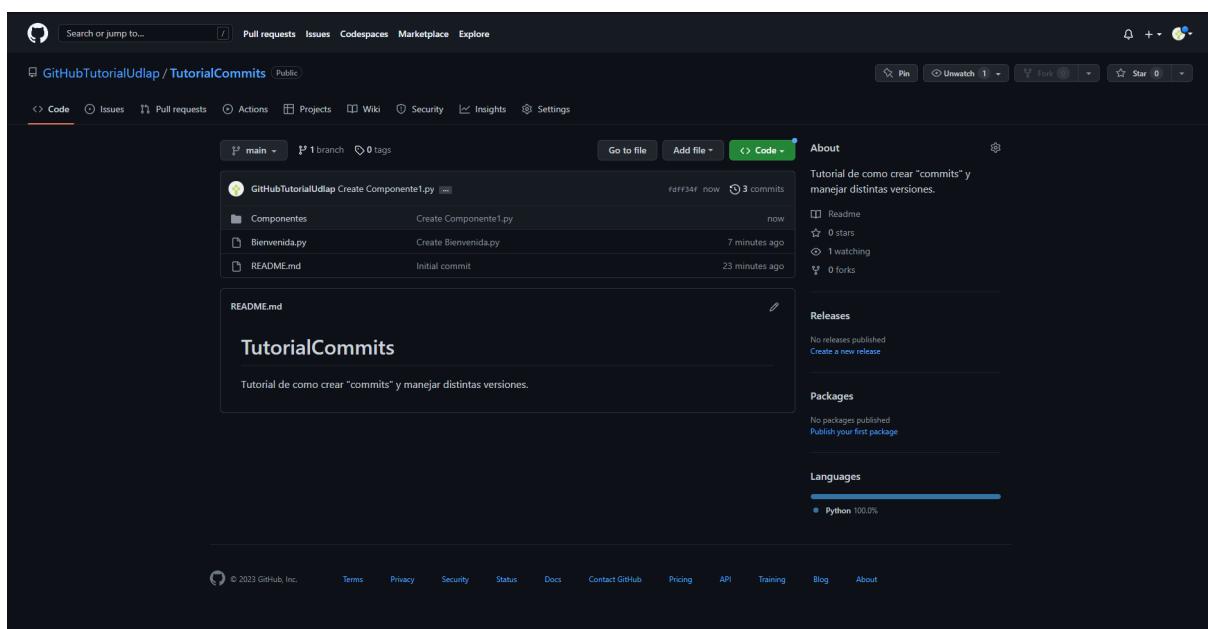
Así se crea un archivo dentro de una nueva carpeta, para sacar al archivo de la carpeta se debe escribir { .. }

En este caso, se puede observar como hay errores de ortografía ya que falta un punto final y errores de sintaxis ya que no están las comillas finales. Es importante saber que GitHub no revisa nada de esto, así que hay que tener cuidado. Los errores se van a arreglar más adelante cuando se hable de distintas versiones.

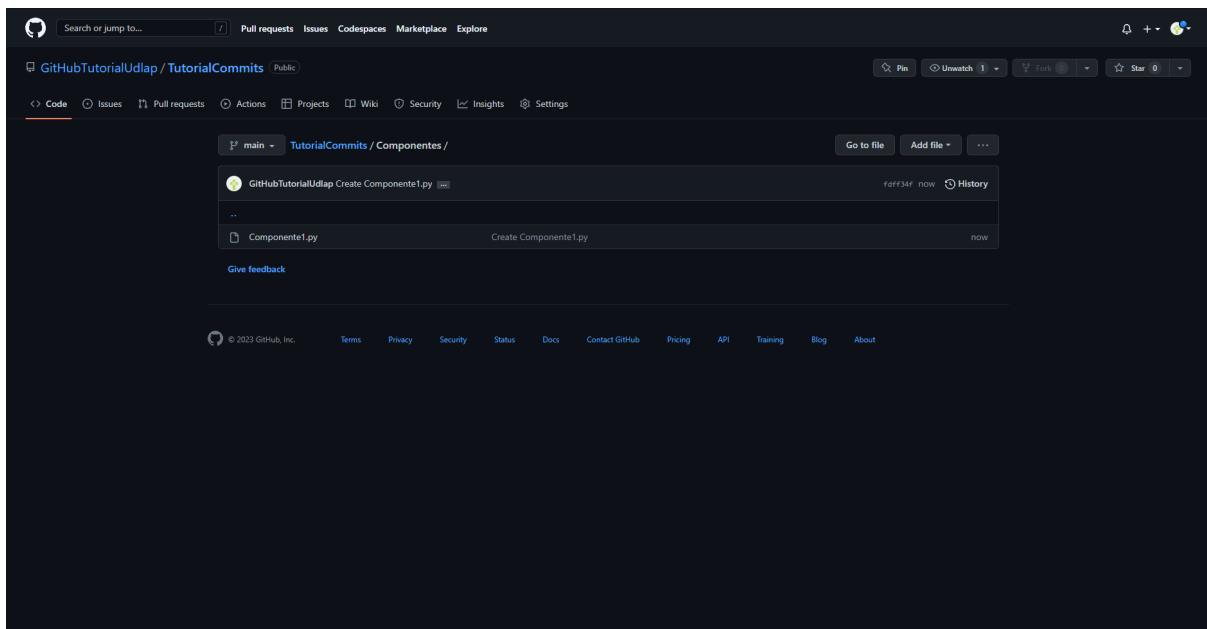
TERCER PASO



No debemos olvidar agregar un mensaje descriptivo antes de presionar el botón "Commit new file".



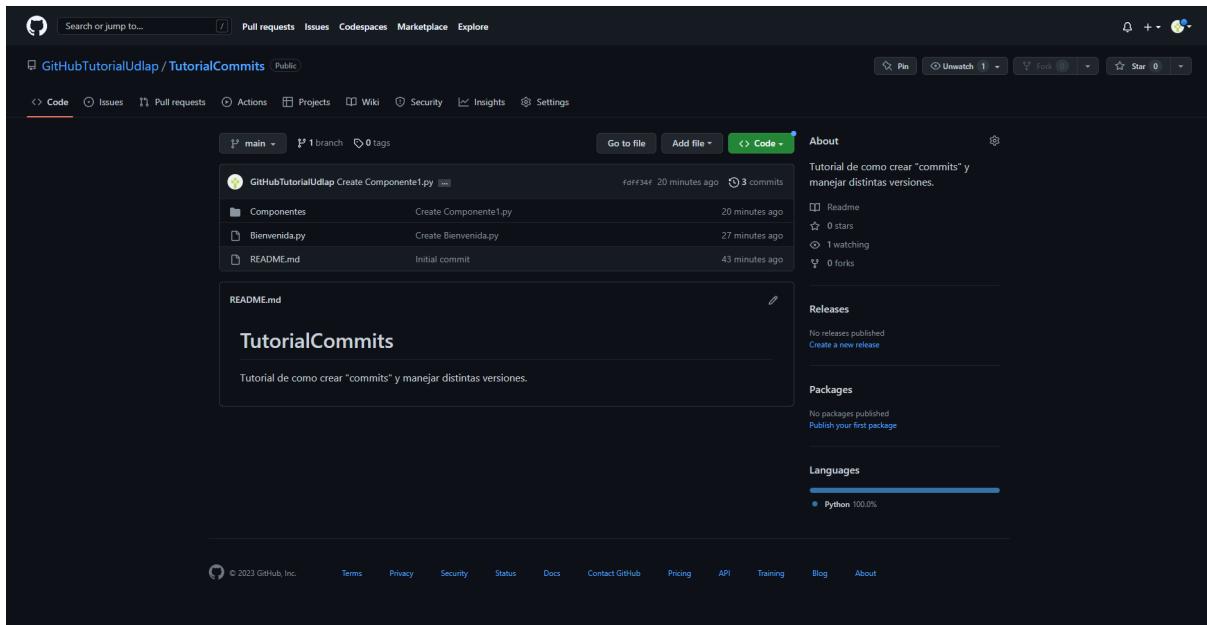
Vista de la creación de la nueva carpeta.



Vista del archivo dentro de la carpeta.

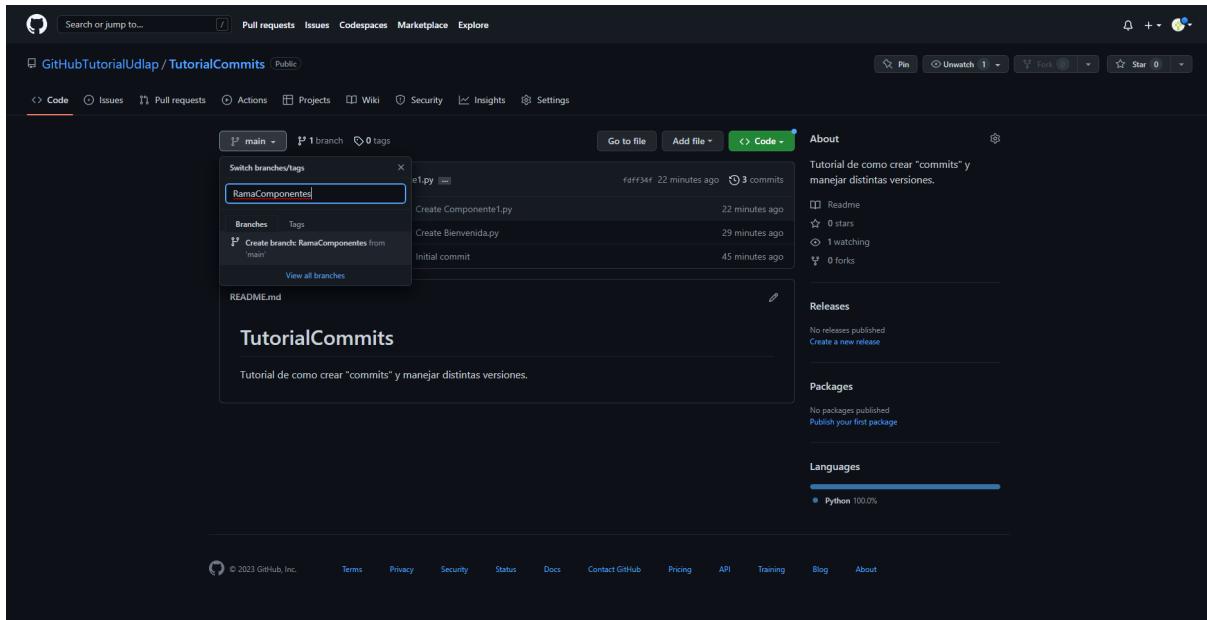
CÓMO MANEJAR LAS DISTINTAS VERSIONES DE UN REPOSITORIO

Caso: Digamos que queremos hacer modificaciones en el documento { Componente1.py } dentro de la carpeta { Componentes }. Queremos agregar una línea de código y además corregir errores de sintaxis y ortografía.

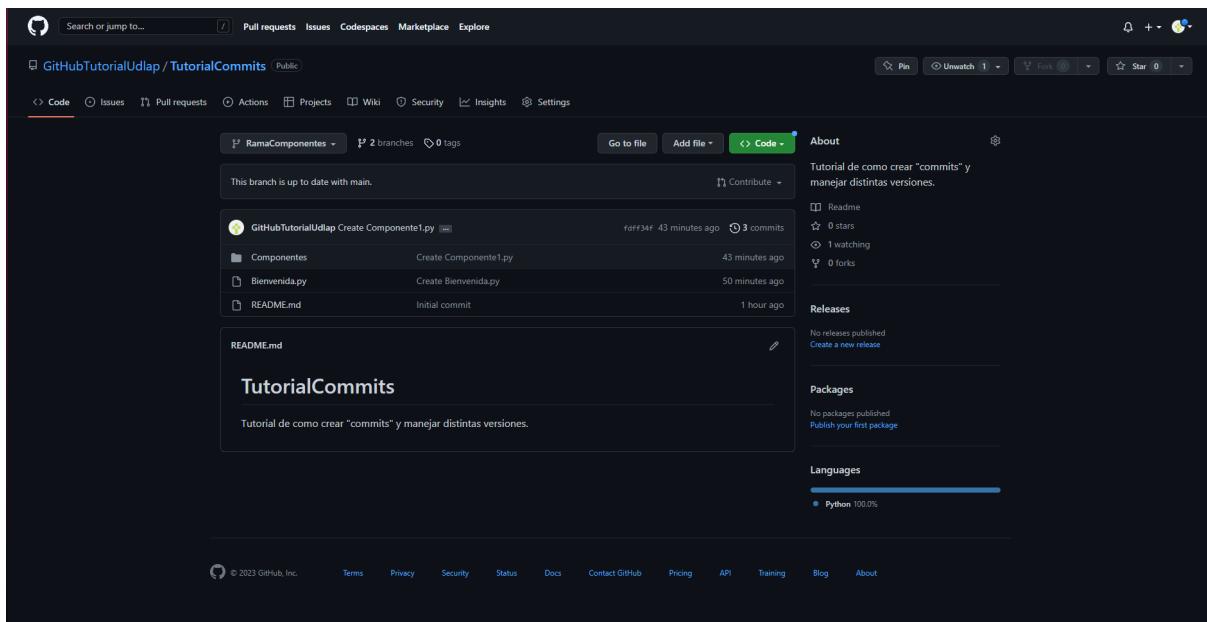


Vista del menú principal del repositorio.

PRIMER PASO



En la vista general hay que ir a donde dice { main } con el icono de las ramas y crear una nueva rama o “branch”. Para esto, es necesario ponerle un nombre descriptivo y sobre todo relacionado a lo que quieras que haga esa rama específica.



En primera instancia parece que no ha sucedido nada, pero ahora podemos ver el siguiente mensaje:



Además, tenemos que la ubicación en la que estamos ya no es { main }, sino { RamaComponentes }. Además, ahora aparece que existen dos ramas en lugar de una:



Sin embargo, ahora realizaremos cambios en la rama y veremos en qué afecta al repositorio. Si entramos al archivo de { Componente1.py } dentro de la carpeta Componentes, podemos ver lo siguiente:

```
print("Este es el primer componente del código")
```

SEGUNDO PASO

```
print("Este es el primer componente del código")
```

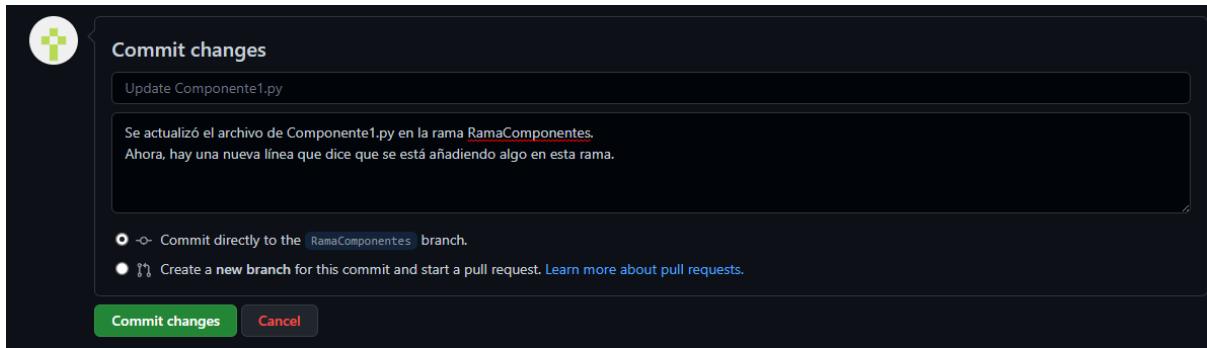
Hacer clic en el botón de editar que tiene forma de lápiz en la parte derecha.

TERCER PASO

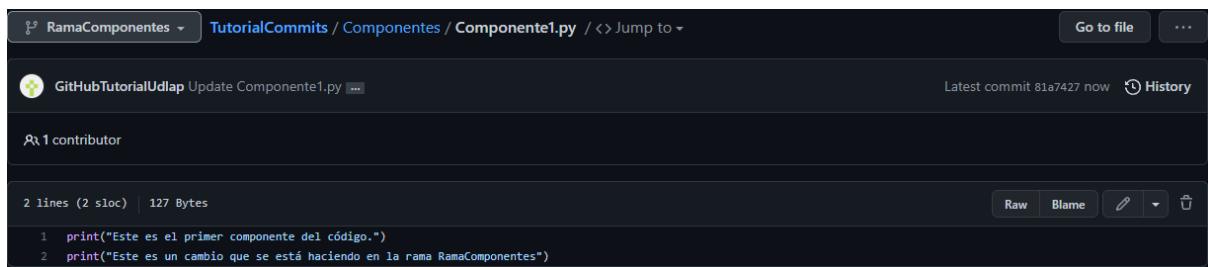
```
print("Este es el primer componente del código.")
print("Este es un cambio que se está haciendo en la rama RamaComponentes.")
```

Agregar un cambio al archivo, en este caso, otra línea de código sencilla y la solución de errores de sintaxis y ortográficos.

CUARTO PASO



Escribir un mensaje descriptivo de lo que se está haciendo. Entre más descriptivo mejor. Ahora podemos observar, que antes donde decía { Commit directly to the main branch }, ahora dice { Commit directly to the RamaComponentes branch }. Lo que es otra indicación de que estamos en otra rama. Después de escribir el mensaje, hacer clic en el botón { Commit changes }. Nota*: ¿Puedes descubrir qué le falta al mensaje para que sea más completo relacionado a los cambios hechos?



Ahora, se ve así nuestra versión del código en RamaComponentes.

Si vamos a la pantalla principal, podemos ver esto:

RamaComponentes had recent pushes 1 minute ago

Compare & pull request

RamaComponentes 2 branches 0 tags Go to file Add file Code

This branch is 1 commit ahead of main.

Contribute

GitHubTutorialUdlap Update Componente1.py ... 81a7427 now 4 commits

Componentes Update Componente1.py now

Bienvenida.py Create Bienvenida.py 1 hour ago

README.md Initial commit 1 hour ago

README.md

TutorialCommits

Tutorial de como crear "commits" y manejar distintas versiones.

Especificamente, el siguiente mensaje es el que nos importa:

RamaComponentes had recent pushes 1 minute ago

Compare & pull request

Sin embargo, si vamos a la misma carpeta pero ahora en la rama principal { main }, podemos observar lo siguiente:

main TutorialCommits / Componentes / Componente1.py / <> Jump to Go to file ...

GitHubTutorialUdlap Create Componente1.py ... Latest commit fdff34f 1 hour ago History

1 contributor

1 lines (1 sloc) | 49 Bytes

1 print("Este es el primer componente del código")

Raw Blame

Los cambios realizados en la rama { RamaComponentes } no se pueden ver en ningún lado. Esto es a lo que se refiere tener distintas versiones del mismo código. Pero ahora queremos llevar los cambios realizados en la rama hacia el archivo principal, ya que es la rama principal.

QUINTO PASO

RamaComponentes had recent pushes 11 minutes ago

Compare & pull request

main 2 branches 0 tags

Go to file Add file Code

GitHubTutorialUdlap Create Componente1.py ... fdff34f 1 hour ago 3 commits

Components Create Componente1.py 1 hour ago

Bienvenida.py Create Bienvenida.py 1 hour ago

README.md Initial commit 1 hour ago

README.md

TutorialCommits

Tutorial de como crear "commits" y manejar distintas versiones.

Volvemos a la pantalla principal del repositorio, pero ahora en la rama { main }. Y presionamos el botón { Compare & pull request } en la parte superior para juntar las ramas.

SEXTO PASO

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: main compare: RamaComponentes Able to merge. These branches can be automatically merged.

Update Componente1.py

Write Preview

Se actualizó el archivo de Componente1.py en la rama RamaComponentes. Ahora, hay una nueva línea que dice que se está añadiendo algo en esta rama. Además, se arreglaron los errores ortográficos y de sintaxis.

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

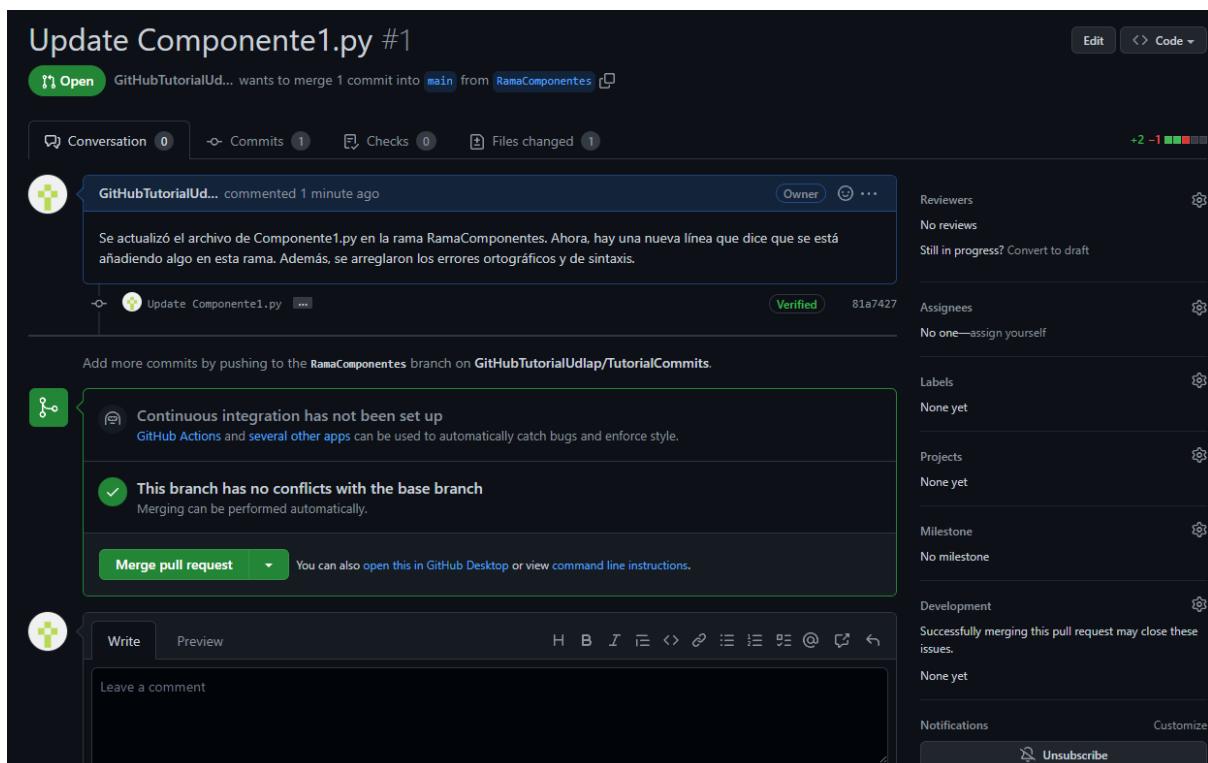
Development: Use Closing keywords in the description to automatically close issues

Helpful resources: GitHub Community Guidelines

-o- 1 commit 1 file changed 1 contributor

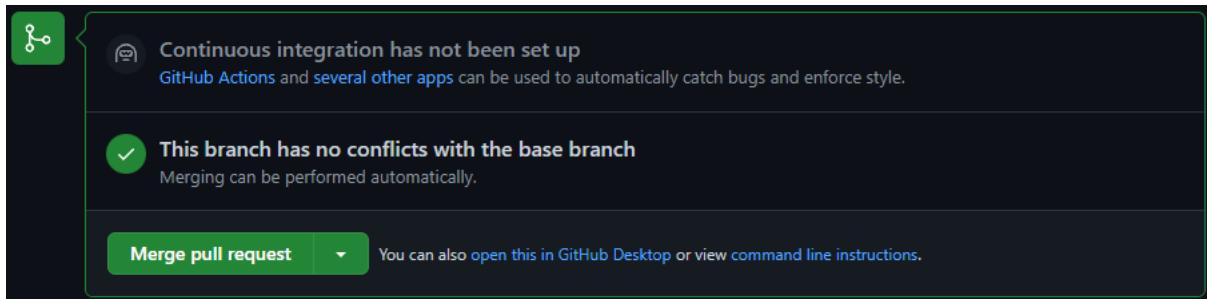
Ahora tenemos esta pantalla. Como podemos ver en el apartado encerrado en rojo. Tenemos un mensaje que dice, de izquierda a derecha, hacia cual rama irán los cambios y desde cual rama vienen. En este caso vendrán de la

{ RamaComponentes } hacia la rama principal { Main }. Además, tenemos una palomilla de color verde que nos dice que estos cambios pueden ser combinados automáticamente. Esto sucede ya que no hicimos una modificación al archivo que tuviera efecto en algo ya existente. Por ejemplo, no borramos una palabra o una línea de código. Por otro lado, en el cuadrado naranja ahora podemos ver y modificar el mensaje que se creó cuando hicimos los cambios en la rama secundaria. La respuesta a la Nota* previamente mencionada es que no se había puesto en el mensaje que se habían corregido los errores ortográficos y de sintaxis. Finalmente, presionamos el botón { Create pull request }.



Ese botón nos lleva a esta pantalla donde podemos hacer varias cosas. Se puede ver cuáles archivos se cambiaron y como, se pueden ver los commits, y se puede ver la conversación, este apartado sirve para realizar últimos cambios. Usualmente esta sección es muy útil al trabajar en equipo o cuando hay varios cambios.

SÉPTIMO PASO



Finalmente, tenemos que juntar el código presionando el botón { Merge pull request } y confirmar esa acción.

The screenshot shows the merged pull request details. It includes a 'Merged' status, commit history, comments from the author, and a success message stating 'Pull request successfully merged and closed'.

Al finalizar la combinación del código. Es posible borrar la rama secundaria, en este caso { RamaComponentes }. También es posible seguir trabajando en ella dependiendo de los requisitos del proyecto. Finalmente, así se ve nuestra nueva rama principal { main }:

The screenshot shows a GitHub repository page for 'TutorialCommits / Componentes / Componente1.py'. The main branch is selected. A single commit from 'GitHubTutorialUdlap' is shown, updating 'Componente1.py'. The commit message is 'Update Componente1.py ...'. It was made 1 hour ago. The code changes are:

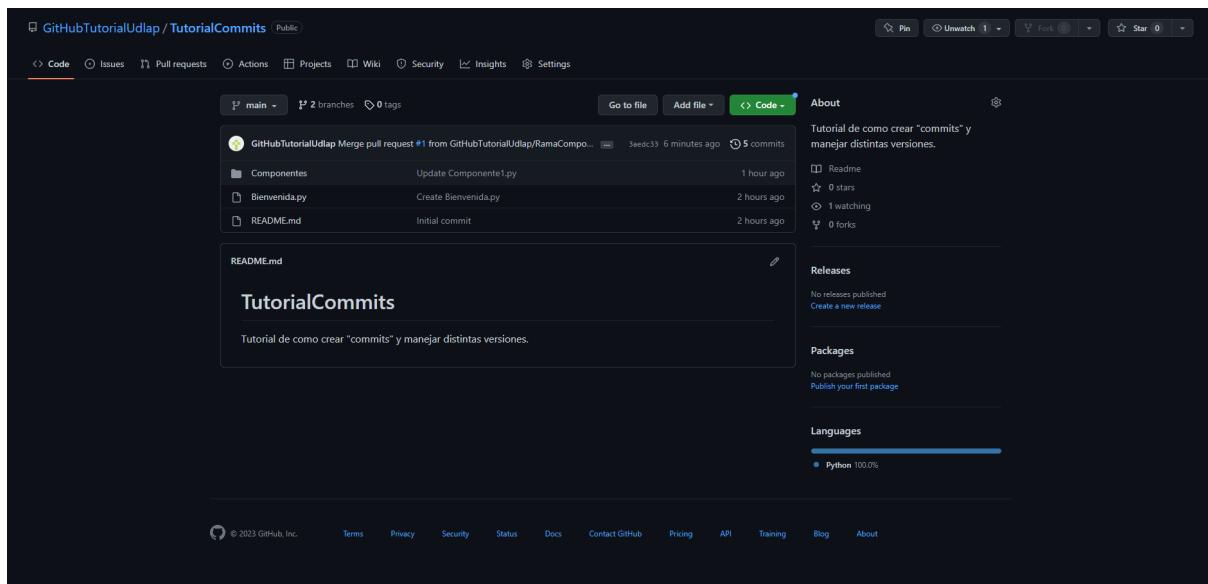
```
1 print("Este es el primer componente del código.")
2 print("Este es un cambio que se está haciendo en la rama RamaComponentes")
```

Below the code, there's a 'Give feedback' button. At the bottom of the page, there are links for GitHub services like Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

Lo que significa que nuestra combinación de versiones fue exitosa.

COMO VER EL HISTORIAL DE VERSIONES

En el caso en el que se quieran revisar cambios específicos en alguna rama secundaria o en la principal, es posible hacerlo siguiendo los siguientes pasos.



Vista de la pantalla de inicio de nuestro repositorio.

PRIMER PASO

This branch is 1 commit behind main.

GitHubTutorialUdlap Update Componente1.py ... 81a7427 1 hour ago 4 commits

File	Commit Message	Time Ago
Componentes	Update Componente1.py	1 hour ago
Bienvenida.py	Create Bienvenida.py	2 hours ago
README.md	Initial commit	3 hours ago

Hacer clic en el botón { 4 commits }. El valor numérico del botón cambiará dependiendo del número de commits que se hayan hecho.

Commits on Jan 22, 2023

Merge pull request #1 from GitHubTutorialUdlap/RamaComponentes ... 3aedc33

GitHubTutorialUdlap committed 14 minutes ago

Update Componente1.py ... 81a7427

GitHubTutorialUdlap committed 1 hour ago

Create Componente1.py ... fdff34f

GitHubTutorialUdlap committed 2 hours ago

Create Bienvenida.py ... 82c5dc0

GitHubTutorialUdlap committed 2 hours ago

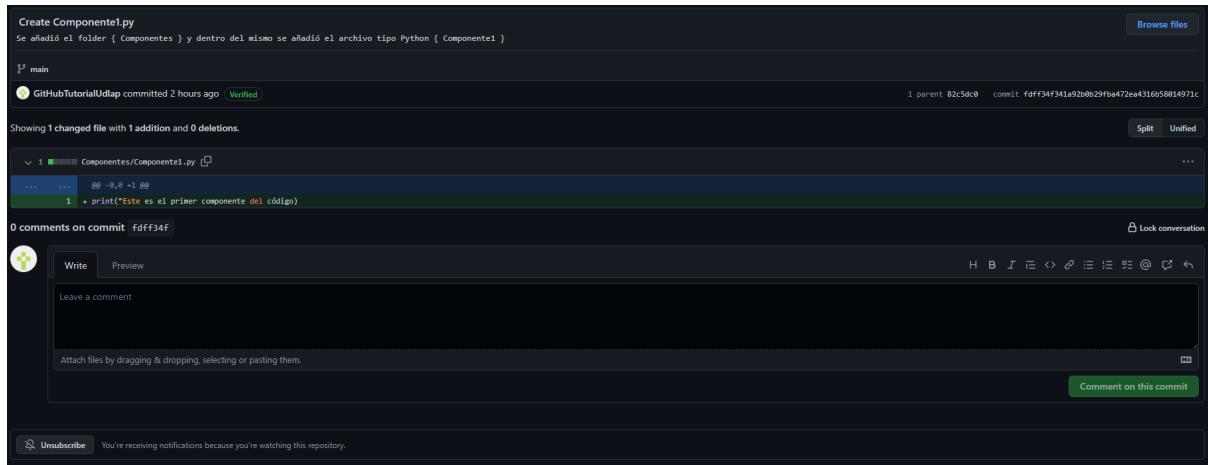
Initial commit 9d349a7

GitHubTutorialUdlap committed 3 hours ago

Newer Older

Aquí podemos observar el historial de versiones de nuestra rama main, se puede cambiar de rama en el botón { main } que se encuentra en la parte superior izquierda. Sin embargo, vamos a ver en qué estado estaba la rama principal antes de combinarla con la rama secundaria. Para esto, se debe hacer clic en el commit { Create Componente1.py }, en el cual si le picamos a los tres puntos a su derecha podemos observar el mensaje que habíamos puesto al hacer el commit. De esta manera el mensaje ayuda, ya que podemos ubicarnos en qué hemos hecho a través de él.

SEGUNDO PASO



Al hacer clic en el commit, este nos lleva a esta pestaña. En ella, podemos ver cómo se veía el código en esta versión, además de que podemos crear comentarios sobre la misma versión y ver hace cuánto tiempo se hizo el commit. Otra cosa importante que se puede observar es quién creó el *commit*. Lo que permite aclarar situaciones de trabajo en equipo con la persona responsable de los cambios.

En caso de querer aprender más de GitHub, favor de entrar a la siguiente liga: <https://docs.github.com/en/get-started>

INTRODUCCIÓN A PROLOG

¿QUÉ ES PROLOG?

Prolog es un lenguaje de programación lógica, utilizado principalmente en proyectos relacionados con inteligencia artificial, como el procesamiento de lenguaje natural. Prolog es un lenguaje declarativo, es decir, de alto nivel (o no procedimental) en el cual el programa especifica que es lo que se tiene que hacer en vez de como hacerlo.

Prolog utiliza un conjunto de relaciones lógicas, similar a SQL. Cuando se corre un programa en Prolog, este es ejecutado por un “motor de inferencia” que responde a una *query* buscando estas relaciones de manera sistemática para hacer inferencias que responderán a dicha *query*.

EL LENGUAJE DE PROLOG

Un programa en Prolog se compone de una serie de hechos (afirmaciones simples) y de reglas (que sirven para afirmar la veracidad de un hecho en base a otros). Estos hechos y reglas se introducen en la base de datos a través de una operación de consulta.

Prolog tiene una sintaxis y semántica simples. Puesto que busca relaciones entre una serie de objetos, la variable y la lista son las estructuras de datos básicas que se usan.

Prolog se basa en un conocimiento base para solucionar consultas así como inferir información. El conocimiento lo representamos mediante cláusulas. Las cláusulas son hechos o reglas, están compuestas usualmente de predicados.

REGLAS

Una regla es del tipo:

Cabeza :- Cuerpo.

y se lee como "La cabeza es verdad si el cuerpo es verdad".

HECHOS

Las cláusulas sin cuerpo son llamados hechos porque siempre son ciertos. Un ejemplo de un hecho es:

gato(tom).

`filosofo(aritoteles).`

`filosofo(platon).`

`fisico(faraday).`

`fisico(feynman).`

`matematico(gauss).`

`matematico(wiles).`

`sabe_matematicas(pepe) :- matematico(pepe).`

`sabe_fisica(pepe) :- fisico(pepe).`

`sabe_filosofia(pepe) :- filosofo(pepe).`



Hechos

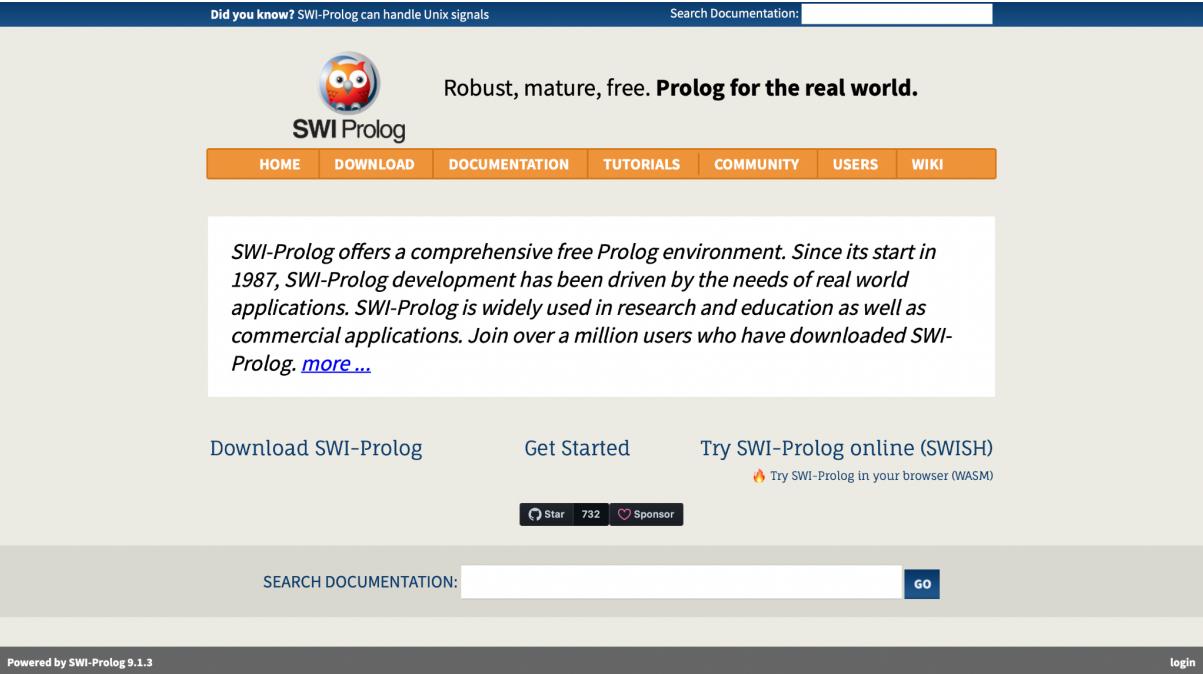


Reglas

Las reglas se pueden “leer” de la siguiente forma: pepe sabe matemáticas si es matemático.

DESCARGA EN MACOS

PRIMER PASO



The screenshot shows the official SWI-Prolog website. At the top, there's a dark header bar with the text "Did you know? SWI-Prolog can handle Unix signals" on the left and a search bar on the right. Below the header is a large banner featuring the SWI-Prolog owl logo and the text "Robust, mature, free. **Prolog for the real world.**". A navigation menu below the banner includes links for HOME, DOWNLOAD, DOCUMENTATION, TUTORIALS, COMMUNITY, USERS, and WIKI. A central text box contains a paragraph about SWI-Prolog's history and usage, ending with a link to "more...". Below this are three calls-to-action: "Download SWI-Prolog", "Get Started", and "Try SWI-Prolog online (SWISH)". The "Get Started" section includes a note about using it in a browser. At the bottom of the page is a footer bar with links for "Powered by SWI-Prolog 9.1.3", "login", and a "GO" button.

Entrar al siguiente link: <https://www.swi-prolog.org>

SEGUNDO PASO

The screenshot shows the official SWI-Prolog website. At the top, there's a blue header bar with the text "Did you know? SWI-Prolog can handle Unix signals" and a search bar labeled "Search Documentation:". Below the header is the SWI-Prolog logo, which is a stylized owl icon, followed by the text "Robust, mature, free. Prolog for the real world." and the word "SWI-Prolog". A navigation menu bar below the logo includes links for HOME, DOWNLOAD, DOCUMENTATION, TUTORIALS, COMMUNITY, USERS, and WIKI. The DOWNLOAD menu item is currently highlighted with an orange background. A large white box contains descriptive text about SWI-Prolog's history and usage, mentioning its start in 1987 and its widespread use in research, education, and commercial applications. Below this box are three calls-to-action: "Download SWI-Prolog", "Get Started", and "Try SWI-Prolog online (SWISH)". There are also GitHub star and sponsorship buttons. A search bar at the bottom is labeled "SEARCH DOCUMENTATION:" with a "GO" button. The footer of the page includes the text "Powered by SWI-Prolog 9.1.3" and a "login" link.

En la parte superior, dar clic en “Download”

TERCER PASO

This screenshot is identical to the one in the second step, showing the SWI-Prolog homepage. The user has clicked on the "Download" link in the navigation menu, which has now turned orange. The rest of the page content, including the main text box, call-to-action buttons, and footer, remains the same.

Esto desplegará una lista. Hacer clic en “SWI-Prolog”

CUARTO PASO

En la parte que dice “Available versions”, dar clic en “Stable release”

Did you know? SWI-Prolog can handle Unix signals

Search Documentation:



SWI-Prolog downloads

- [HOME](#)
- [DOWNLOAD](#)
- [DOCUMENTATION](#)
- [TUTORIALS](#)
- [COMMUNITY](#)
- [USERS](#)
- [WIKI](#)

Available versions

The **stable** release is infrequently updated. It is fine for running basic Prolog code without surprises. The **development** version is released roughly every two to four weeks. This is the recommended version for developers and users of applications such as [SWISH](#) or [ClipPatricia](#). Finally, the **GIT** and **daily** versions are for developers that want to contribute or have immediate access to patches. These versions are generally fine, but occasionally suffer from regression.

- [Stable release](#)
- [Development release](#)
- [Daily builds for Windows](#)
- [Browse GIT repository](#)

Read more about

- Available SWI-Prolog [versions](#)
- Information on [Linux packages and building on Linux](#)

Tags: [Linux](#) [MacOSX](#) [Windows](#) [swi-prolog/Download](#) [windows](#) Tag Wiki page "SWI-Prolog downloads"

LogicalCaptain said (2020-05-22T08:44:11):

Note

Tags are associated to your profile if you are logged in|Report abuse

QUINTO PASO

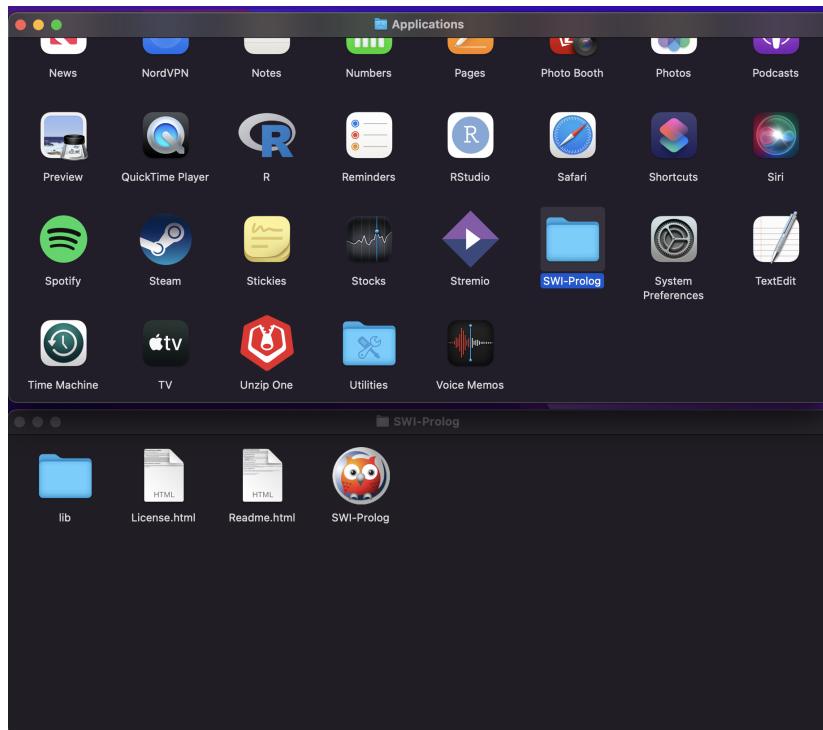
Seleccionar el build que aplique para su Mac. Debe ser uno de estos dos:

	51,732,645 bytes	SWI-Prolog 9.0.3-1 for MacOSX 10.14 (Mojave) and later on x86_64 and arm64 Installer with binaries created using Macports . Installs /opt/local/bin/swipl. Needs xquartz (X11) and the Developer Tools (Xcode) installed for running the development tools SHA256: 1e5f3a86ba52823833ecb21837fc2381ba6fef04c9bb540ed1671fad23443de
	28,195,489 bytes	SWI-Prolog 8.4.1-1 for MacOSX bundle on intel Installer with binaries created using Macports . Installs /opt/local/bin/swipl. Needs xquartz (X11) and the Developer Tools (Xcode) installed for running the development tools SHA256: 1b9c62caa781818a0dafd1d822ab563b8c10c7cd018ce10a3b71f900eb3a434f

CONFIGURACIÓN EN MACOS

PRIMER PASO

Una vez instalado, se recomienda crear un folder dentro de “Applications”. Arrastrar todos los archivos dentro del build que se acaba de descargar a dicho folder. Se tiene que ver algo así:



SEGUNDO PASO

Para abrir Prolog, mantener apretado el botón “control” y hacer clic con el mouse. Dar clic en “Open”. Se mostrará una pantalla diciendo que no se puede verificar la aplicación. Dar clic en “Open” y se abrirá esta pestaña:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
```

Esto significa que Prolog fue instalado correctamente.

DESCARGA EN WINDOWS

PRIMER PASO

The screenshot shows the official SWI-Prolog website. At the top, there's a dark header bar with the text "Did you know? SWI-Prolog can handle Unix signals" on the left and a search bar on the right. Below the header is the SWI-Prolog logo, which features a cartoon owl icon and the text "SWI Prolog". A tagline "Robust, mature, free. Prolog for the real world." is displayed next to the logo. A navigation menu bar below the logo contains links for "HOME", "DOWNLOAD", "DOCUMENTATION", "TUTORIALS", "COMMUNITY", "USERS", and "WIKI". A central text box contains a paragraph about SWI-Prolog's history and user base, ending with a link to "more...". At the bottom of the page, there are three calls-to-action: "Download SWI-Prolog", "Get Started", and "Try SWI-Prolog online (SWISH)". There's also a note about trying it in a browser using WASM.

Entrar al siguiente link: <https://www.swi-prolog.org>

SEGUNDO PASO

This screenshot shows the same SWI-Prolog website as above, but with the "DOWNLOAD" menu item from the navigation bar highlighted. A dropdown menu has appeared, listing "SWI-Prolog", "Sources/building", "Docker images", "Add-ons", and "Browse GIT". The main content area of the page remains the same, displaying the introductory text about the SWI-Prolog environment and its history.

En la parte superior, hacer clic en “Download”. Despu s hacer clic en “SWI-Prolog”.

TERCER PASO

Did you know? SWI-Prolog can handle Unix signals Search Documentation:

 **SWI-Prolog downloads**

HOME DOWNLOAD DOCUMENTATION TUTORIALS COMMUNITY USERS WIKI

Available versions

The **stable** release is infrequently updated. It is fine for running basic Prolog code without surprises. The **development** version is released roughly every two to four weeks. This is the recommended version for developers and users of applications such as [SWISH](#) or [ClioPatria](#). Finally, the **GIT** and **daily** versions are for developers that want to contribute or have immediate access to patches. These versions are generally fine, but occasionally suffer from regression.

- [Stable release](#)
- [Development release](#)
- [Daily builds for Windows](#)
- [Browse GIT repository](#)

Read more about

- Available SWI-Prolog [versions](#)
- Information on [Linux packages and building on Linux](#)

En “Available versions”, hacer clic en “Stable release”.

CUARTO PASO

Binaries		
	13,156,103 bytes	SWI-Prolog 9.0.3-1 for Microsoft Windows (64 bit) Self-installing executable for Microsoft's Windows 64-bit editions. Requires at least Windows 7. See the reference manual for deciding on whether to use the 32- or 64-bits version. This binary is linked against GMP 6.1.1 which is covered by the LGPL license. SHA256: caf7c68d5095845a8a310b3297721abf6465bc599a814efcfe831535ea546e29
	13,195,424 bytes	SWI-Prolog 9.0.3-1 for Microsoft Windows (32 bit) Self-installing executable for MS-Windows. Requires at least Windows 7. Installs swipl-win.exe and swipl.exe . This binary is linked against GMP 6.1.1 which is covered by the LGPL license. SHA256: 50e63e1cece1e006994e640f13f68f00448c0194532990ad549676c6edc4c4d6
	51,732,645 bytes	SWI-Prolog 9.0.3-1 for MacOSX 10.14 (Mojave) and later on x86_64 and arm64 Installer with binaries created using Macports . Installs /opt/local/bin/swipl. Needs xquartz (X11) and the Developer Tools (Xcode) installed for running the development tools SHA256: 1e5f3a80e0a52823839ec21837fc2381bafe04c9bb540ed1671f1ad23443de
	28,195,489 bytes	SWI-Prolog 8.4.1-1 for MacOSX bundle on intel Installer with binaries created using Macports . Installs /opt/local/bin/swipl. Needs xquartz (X11) and the Developer Tools (Xcode) installed for running the development tools SHA256: 1b9c62caa781818a0daef1d1822aab563b8c10c7cd018ce1a3b71f900eb3a434f
Sources		
	11,827,064 bytes	SWI-Prolog source for 9.0.3 Sources in .tar.gz format, including packages and generated documentation files. See build instructions . SHA256: e2919bc58710abd62b9cd40179a724c3b0dbe9ae428af49d7fdcd60158921afb
Documentation		
	3,140,650 bytes	SWI-Prolog 9.0.3 reference manual in PDF SWI-Prolog reference manual as PDF file. This does <i>not</i> include the package documentation. Show all files

Seleccionar el build que aplique para su computadora.

	13,156,103 bytes	SWI-Prolog 9.0.3-1 for Microsoft Windows (64 bit) Self-installing executable for Microsoft's Windows 64-bit editions. Requires at least Windows 7. See the reference manual for deciding on whether to use the 32- or 64-bits version. This binary is linked against GMP 6.1.1 which is covered by the LGPL license. SHA256: caf7c68d5095845a8a310b3297721abf6465bc599a814efcfe831535ea546e29
	13,195,424 bytes	SWI-Prolog 9.0.3-1 for Microsoft Windows (32 bit) Self-installing executable for MS-Windows. Requires at least Windows 7. Installs swipl-win.exe and swipl.exe . This binary is linked against GMP 6.1.1 which is covered by the LGPL license. SHA256: 50e63e1cece1e006994e640f13f68f00448c0194532990ad549676c6edc4c4d6

Para Windows debe ser uno de estos dos.

QUINTO PASO

⚠️ Windows antivirus software works using *signatures* and *heuristics*. Using the huge amount of viruses and malware known today, arbitrary executables are often [falsely classified as malicious](#). [Google Safe Browsing](#), used by most modern browsers, therefore often classifies our Windows binaries as malware. You can use e.g., [virustotal](#) to verify files with a large number of antivirus programs.

Our Windows binaries are cross-compiled on an isolated Linux container. The integrity of the binaries on the server is regularly verified by validating its SHA256 fingerprint.

Please select the checkbox below to enable the actual download link.

I understand

[Download swipl-9.0.3-1.x64.exe](#) (SHA256: caf7c68d5095845a8a310b3297721abf6465bc599a814efcf831535ea546e29)

[VIRUSTOTAL Scan Result](#)

Hacer clic sobre el recuadro para aceptar que entendemos este aviso sobre que el programa puede ser detectado como un virus por el antivirus de Windows.

SEXTO PASO

⚠️ Windows antivirus software works using *signatures* and *heuristics*. Using the huge amount of viruses and malware known today, arbitrary executables are often [falsely classified as malicious](#). [Google Safe Browsing](#), used by most modern browsers, therefore often classifies our Windows binaries as malware. You can use e.g., [virustotal](#) to verify files with a large number of antivirus programs.

Our Windows binaries are cross-compiled on an isolated Linux container. The integrity of the binaries on the server is regularly verified by validating its SHA256 fingerprint.

Please select the checkbox below to enable the actual download link.

I understand

[Download swipl-9.0.3-1.x64.exe](#) (SHA256: caf7c68d5095845a8a310b3297721abf6465bc599a814efcf831535ea546e29)

[VIRUSTOTAL Scan Result](#)

Hacer clic en el recuadro para que se habilite el enlace para descargar el programa.

SÉPTIMO PASO



Hacer clic en el archivo descargado.

OCTAVO PASO



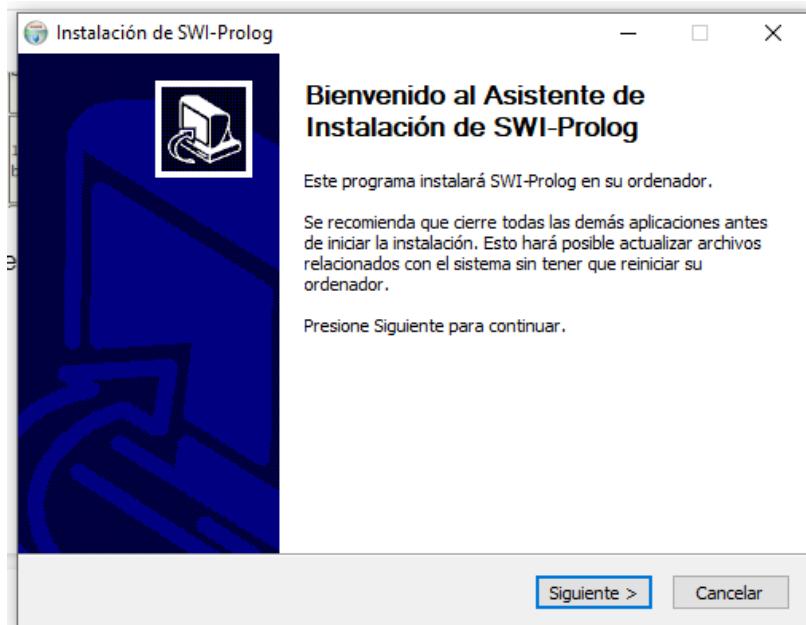
Hacer clic en Más información.

NOVENO PASO



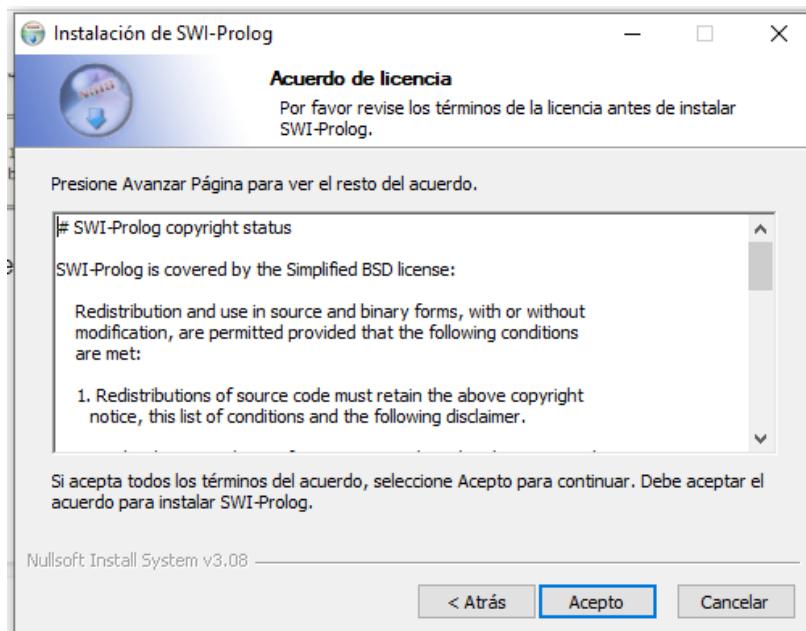
Hacer clic en Ejecutar de todas formas.

DÉCIMO PASO



Hacer clic en Siguiente para iniciar con la instalación del programa.

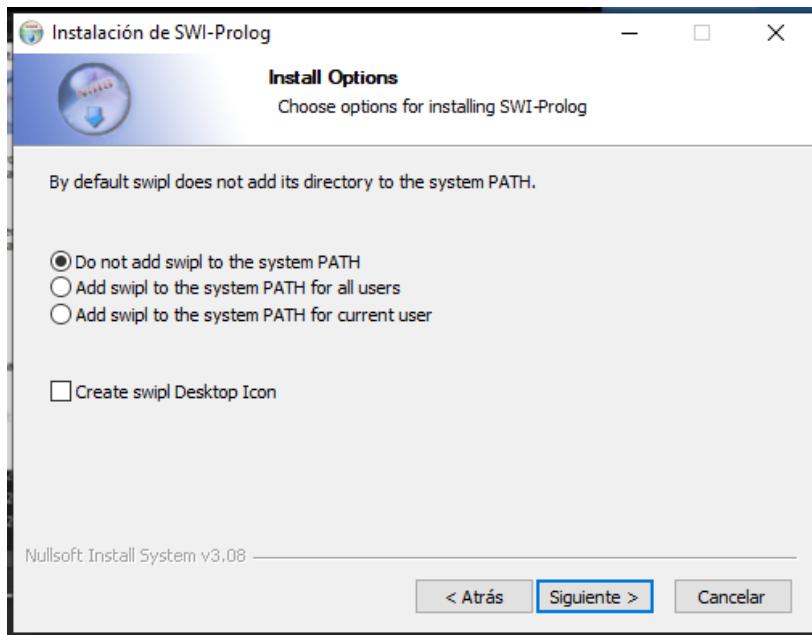
UNDÉCIMO PASO



Hacer clic en Aceptar para aceptar el acuerdo de licencia del programa.

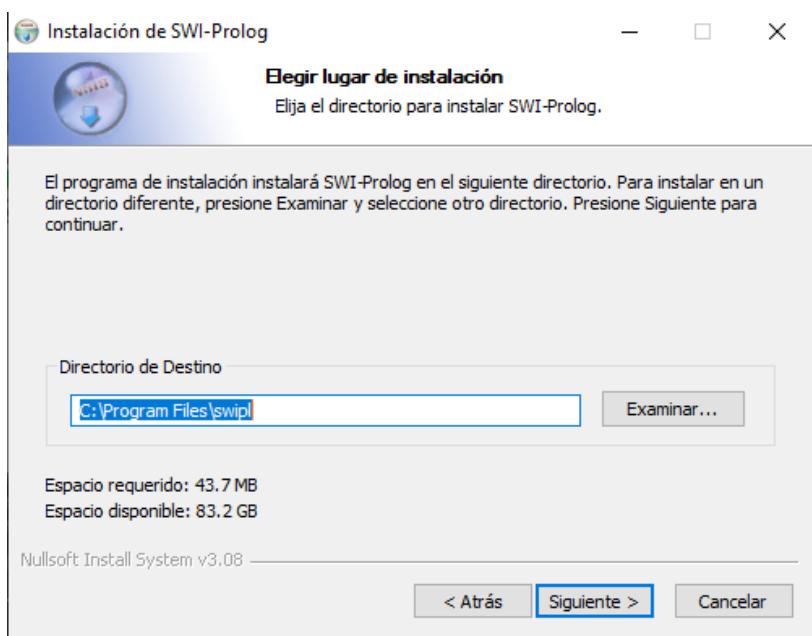
OJO: A partir de aquí la configuración del programa puede ser modificada a nuestro antojo. Nosotros dejamos la configuración por defecto.

DUODÉCIMO PASO



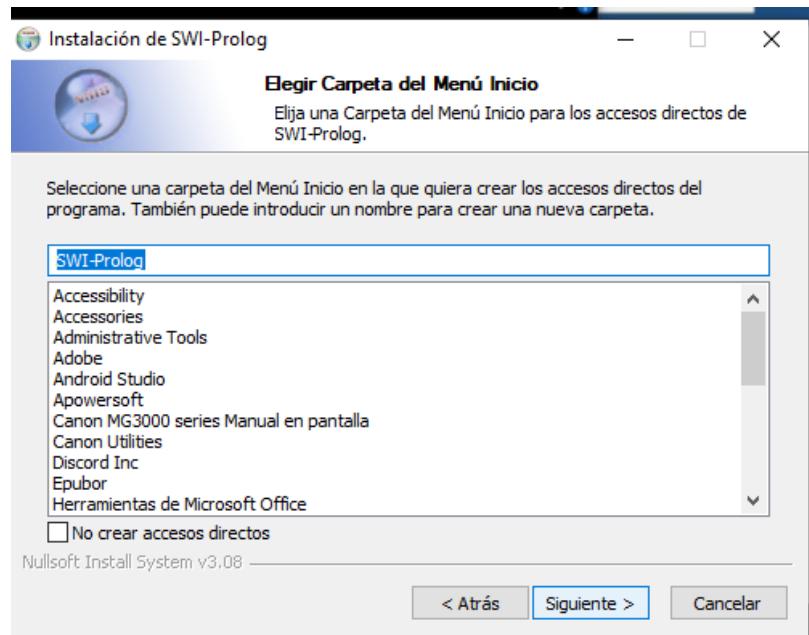
Seleccionar la opción deseada sobre si se quiere o no agregar el directorio a la ruta del sistema. Después hacer clic en Siguiente.

DECIMOTERCER PASO



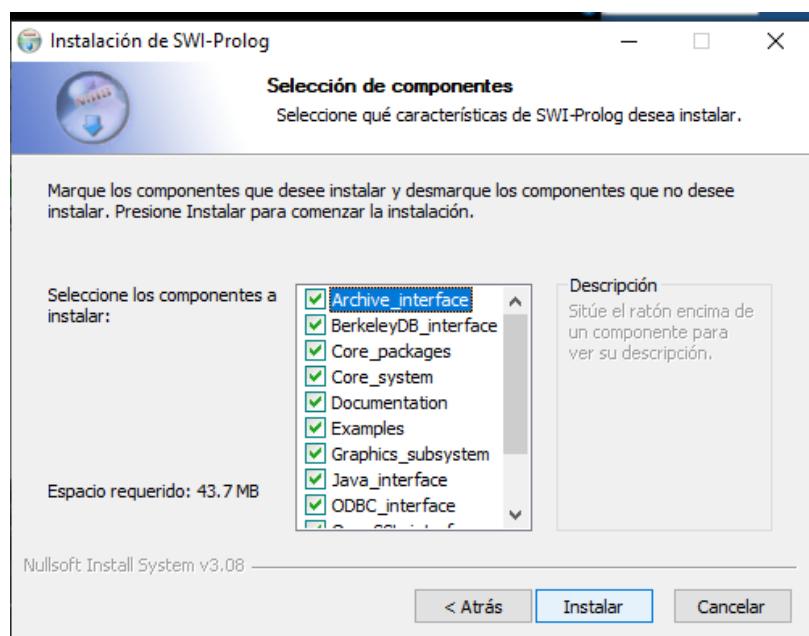
Seleccionar la carpeta en donde se instalará el programa. Después hacer clic en Siguiente.

DECIMOCUARTO PASO



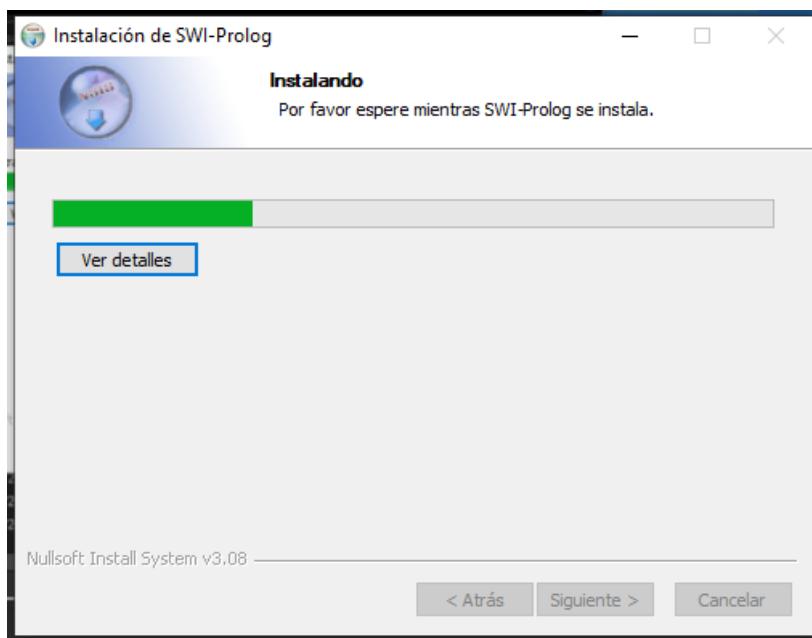
Seleccionar la carpeta del Menú Inicio en la que se quieran crear los accesos directos. Después hacer clic en Siguiente.

DECIMOQUINTO PASO



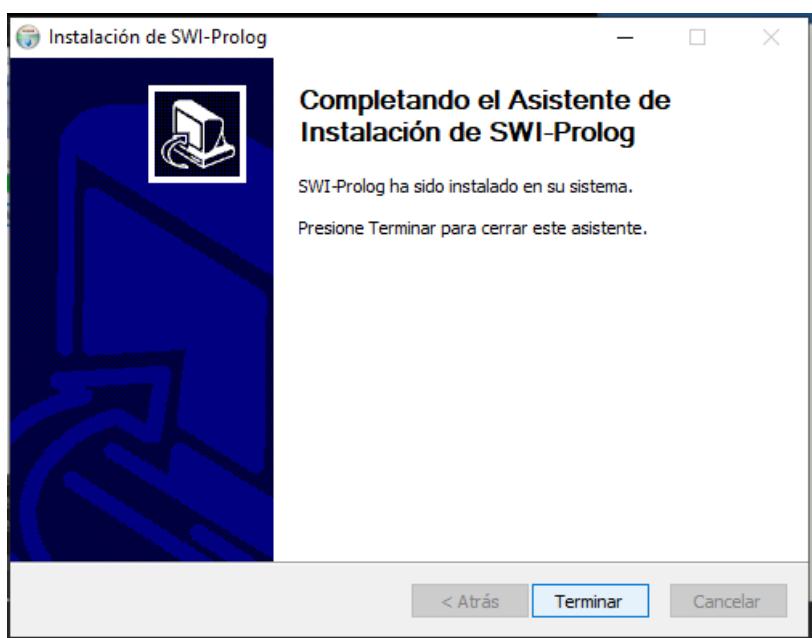
Seleccionar los componentes que deseamos instalar o deseleccionar los que no deseamos instalar. Después hacer clic en Instalar.

DECIMOSEXTO PASO



Esperar a que finalice la instalación.

DECIMOSÉPTIMO PASO



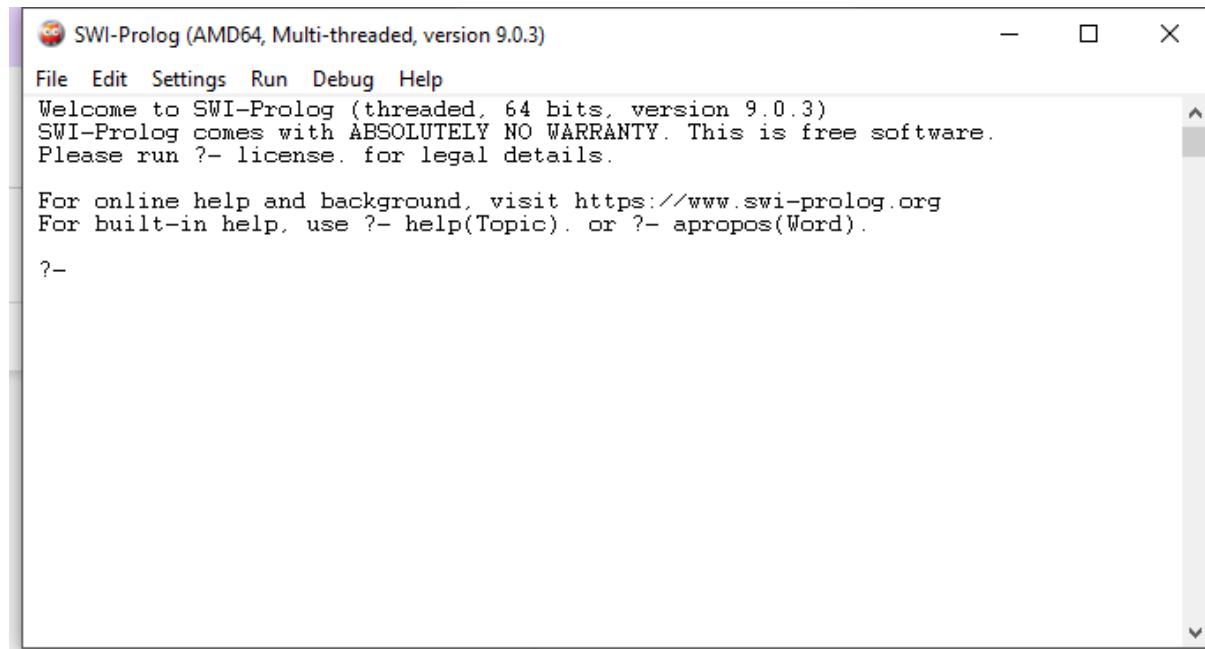
Hacer clic en Terminar.

Se ha instalado Prolog correctamente.

USO DE PROLOG

PRIMER PASO

Abrir Prolog.

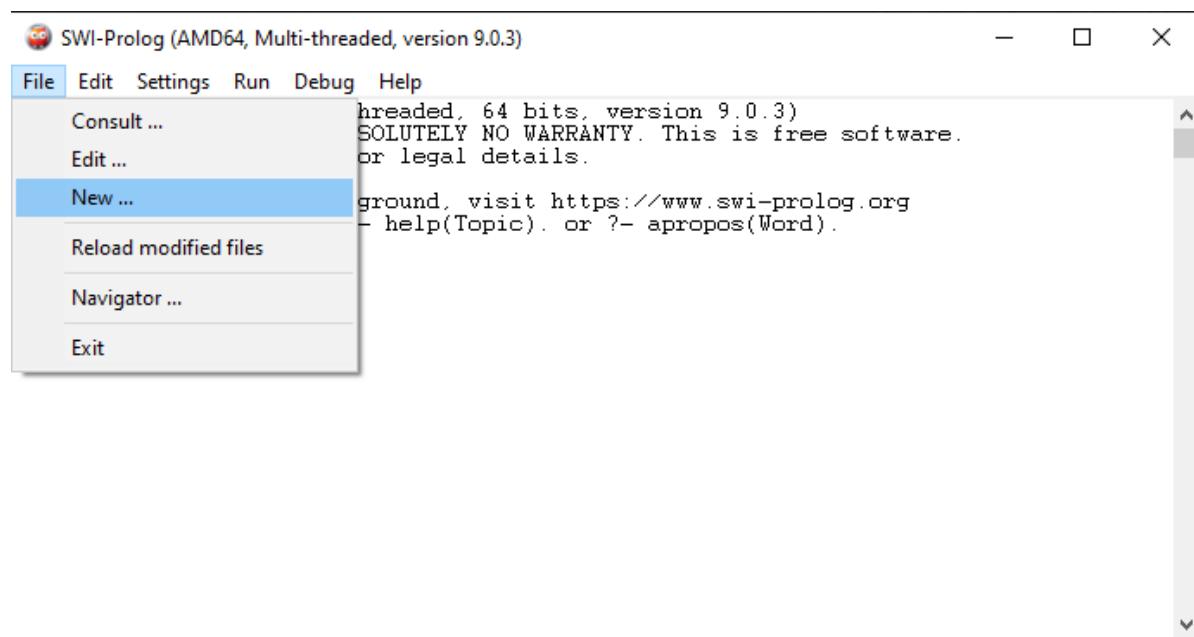


Al abrir Prolog aparece esta ventana.

Vemos la consola Prolog y el prompt, que espera un comando para ejecutar.

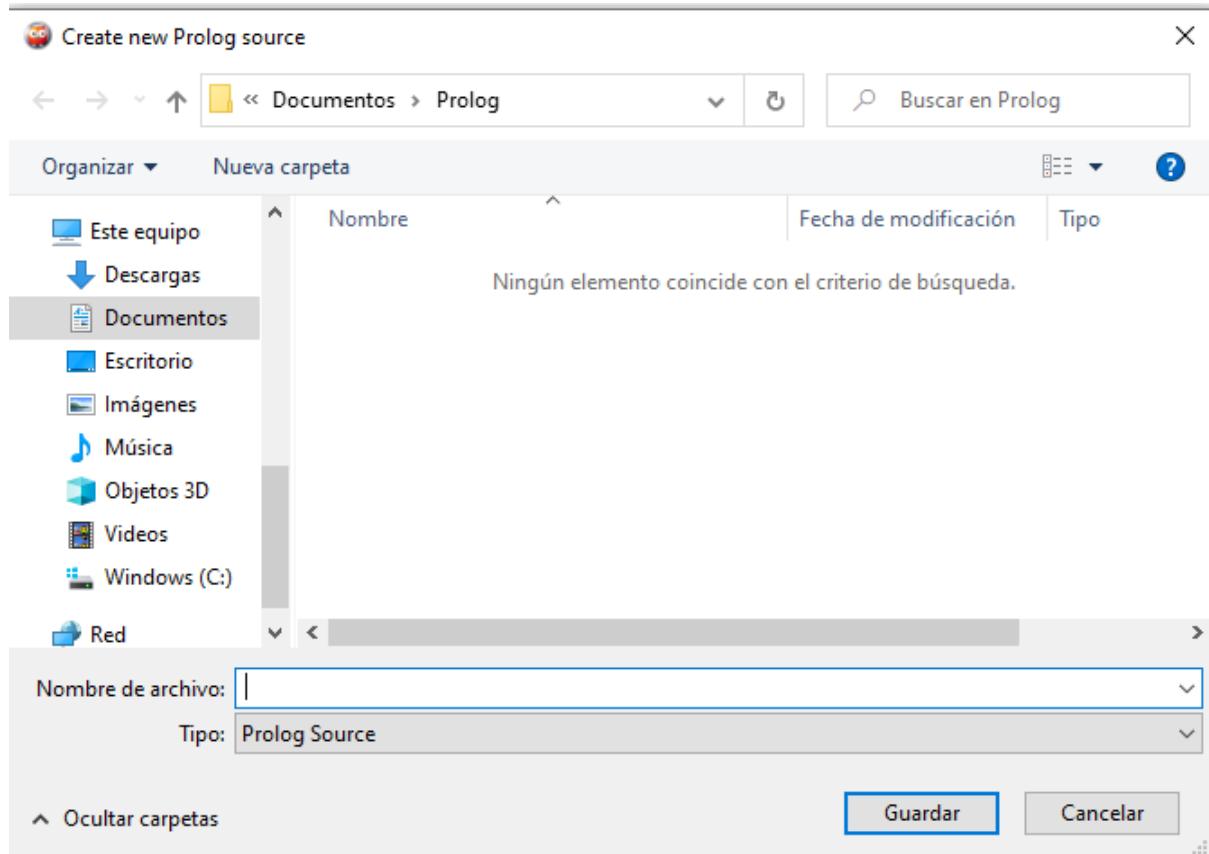
CREAR NUEVO PROYECTO

PRIMER PASO



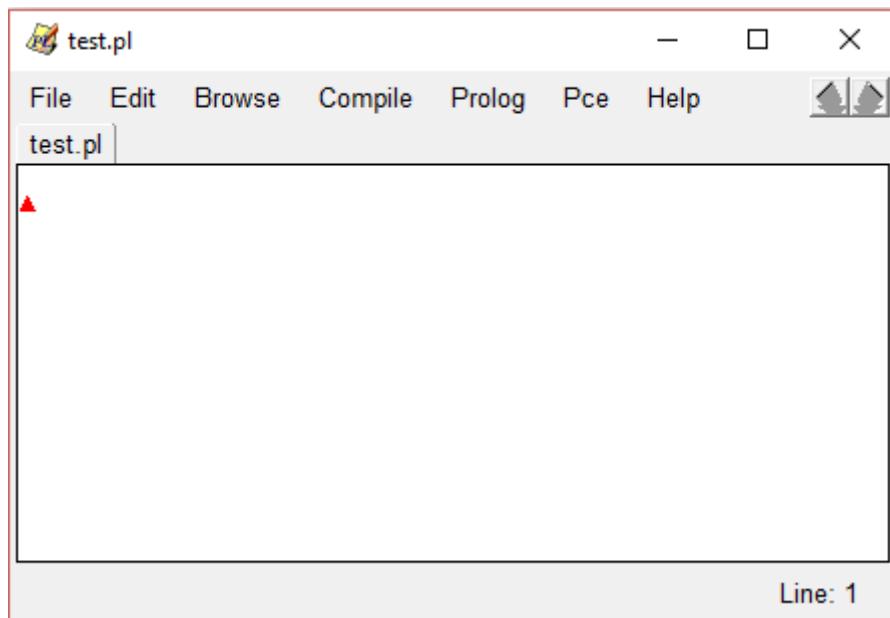
Hacemos clic en “File” y después en “New”.

SEGUNDO PASO



Seleccionar la ubicación del archivo.

TERCER PASO



Se abre otra ventana que contendrá las reglas y los predicados del archivo. Escribimos el contenido del motor.

EJEMPLO DE PROLOG

CUARTO PASO

```
File Edit Browse Compile Prolog Pce Help
test.pl [modified]
comer(gato, raton).
comer(raton, queso).

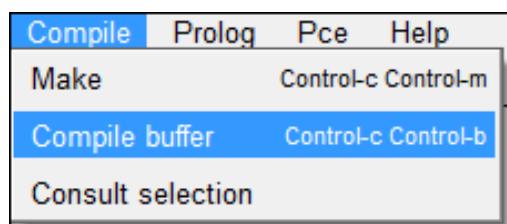
piel(gato).
piel(raton).

amigos(X, Y) :-
    comer(X, Z),
    comer(Z, Y).▲

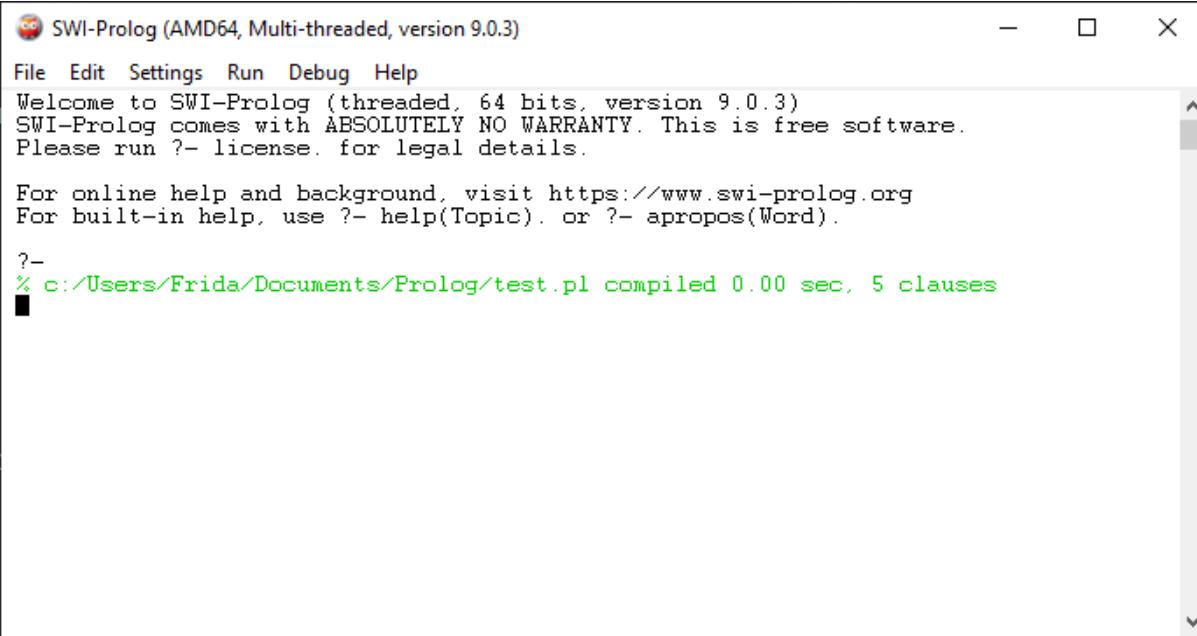
Colourising buffer ... done, 0.00 seconds, 14 fragments
Line: 1
```

Escribir el ejemplo anterior. Guardar el archivo.

QUINTO PASO



Compilar. Hacer clic en “Compile” y después en “Compile buffer”.



The screenshot shows the SWI-Prolog IDE interface. The title bar reads "SWI-Prolog (AMD64, Multi-threaded, version 9.0.3)". The menu bar includes File, Edit, Settings, Run, Debug, and Help. A welcome message from the software is displayed, stating: "Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.3). SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software. Please run ?- license. for legal details." Below this, instructions for online help and built-in help are provided. The main window shows the command line: "?- % c:/Users/Frida/Documents/Prolog/test.pl compiled 0.00 sec, 5 clauses".

En la consola, debe aparecer un mensaje indicando que se ha compilado el archivo. De lo contrario, corregir los errores y repetir la operación.

SEXTO PASO

```
1 ?- piel(X).
X = gato ;
X = raton.

2 ?- comer(gato,X).
X = raton.

3 ?- amigos(X,Y).
X = gato,
Y = queso ;
false.

4 ?- piel(queso).
false.

5 ?-
```

Hacer las consultas anteriores.

Bibliografía

Ediciones Eni (s.f.) *Uso de SWI-Prolog*. Recuperado el 24 de enero de 2023 de:

https://www.ediciones-eni.com/open/mediabook.aspx?idR=871cdd90cb_c56128cec45c0e9a030694

Encyclopedi a Britannica. (s. f.). *Computer programming language - Visual Basic*. Recuperado el 22 de enero de 2023 de:
<https://www.britannica.com/technology/computer-programming-language/Visual-Basic#ref849838>

GeeksforGeeks. (2018). *Prolog | An Introduction*. Recuperado el 22 de enero de 2023 de: <https://www.geeksforgeeks.org/prolog-an-introduction/>

GitHub Docs. (s.f.). *GitHub glossary*. Recuperado el 22 de enero de 2023 de:
<https://docs.github.com/en/get-started/quickstart/github-glossary>

Muñoz, A. (2022). *¿Qué es GitHub y para qué sirve?* Recuperado el 23 de enero de 2023 de:
<https://www.webempresa.com/hosting/que-es-github.html>

SWI-Prolog. (s.f.). *SWI-Prolog*. Recuperado el 23 de enero de 2023 de:
<https://www.swi-prolog.org>

Downloading & Setting Up SWI-Prolog (MacOS). (n.d.). [Www.youtube.com. Retrieved June 5, 2022, from https://www.youtube.com/watch?v=KZJDEiFvABk](https://www.youtube.com/watch?v=KZJDEiFvABk)