



## Ciencias de la Computación

**Objetivo:** Este documento está diseñado para que los alumnos de nuevo ingreso a la carrera de Ciencias de la Computación de la Facultad de Ciencias de la UNAM, puedan tener una guía básica de los elementos con los que debe contar para iniciar con el aprendizaje en la carrera.

En esta sección haremos un listado de diversos requisitos que se necesitan tener en un ordenador para empezar a programar. También aprenderás a instalar paquetes a través de una terminal, es decir, con comandos propios de Linux.

## Índice

<b>1. ¿Cómo instalar Java?</b>	<b>2</b>
1.1. Instalación en Ubuntu . . . . .	2
1.2. Instalación en Fedora . . . . .	3
<b>2. Instalando el compilador de Java</b>	<b>5</b>
<b>3. Variables de entorno de Java</b>	<b>6</b>
<b>4. ¿Cómo instalar un editor de textos?</b>	<b>8</b>
4.1. Sublime Text 3 . . . . .	8
4.1.1. Instalación en Ubuntu . . . . .	8
4.1.2. Instalación en Fedora . . . . .	10
4.2. Visual Studio Code . . . . .	11
4.2.1. Instalación en Ubuntu . . . . .	12
4.2.2. Instalación en Fedora . . . . .	13
4.3. Emacs . . . . .	14
4.3.1. Instalación en Ubuntu . . . . .	14
4.3.2. Instalación en Fedora . . . . .	15
4.4. Vim . . . . .	16
4.4.1. Instalación en Ubuntu . . . . .	16
4.4.2. Instalación en Fedora . . . . .	17
<b>5. Introducción a GitHub;</b>	<b>19</b>
5.1. ¿Cómo crear una cuenta en GitHub? . . . . .	19
5.2. ¿Cómo crear un repositorio en GitHub? . . . . .	24
5.3. ¿Cómo subir un proyecto a GitHub? . . . . .	27
5.4. ¿Cómo generar un token en GitHub? . . . . .	33
5.5. ¿Cómo clonar un repositorio y subir cambios en GitHub? . . . . .	37
5.6. ¿Cómo crear ramas y cambiarlas en GitHub? . . . . .	42
5.7. ¿Cómo descargar cambios en GitHub? . . . . .	47
5.8. ¿Cómo mezclar varias ramas en un proyecto? . . . . .	49
5.9. ¿Cómo eliminar una rama en GitHub? . . . . .	52



## 1. ¿Cómo instalar Java?

Java es el primer lenguaje de programación que se utiliza para enseñar a programar en la Facultad de Ciencias por lo que es de vital importancia tenerlo instalado en nuestra computadora o máquina virtual.

Este lenguaje de programación se instala a través de comandos que hay que ejecutar en la consola de Linux y podemos instalar la versión que mejor nos acomode. En este caso, instalaremos [Java 17](#).

**Nota:** Antes de hacer la instalación pregunta a tus profesores qué versión estarán usando en el curso para que no tengas ningún problema con ello.

### 1.1. Instalación en Ubuntu

Primero checamos qué versión de Ubuntu tenemos instalado. Abrimos nuestra terminal Ctrl+Alt+T y ejecutamos el siguiente comando:

```
$ lsb_release -a
```

Nos da la siguiente salida:

```
fciencias@fciencias-VirtualBox:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.10
Release:        22.10
Codename:       kinetic
fciencias@fciencias-VirtualBox:~$
```

Tenemos instalado Ubuntu 22.10, ahora ejecutamos el siguiente comando para ver las versiones de java que tenemos disponibles.

```
$ java -version
```

Nos aparecerá lo siguiente pues no lo tenemos instalado.

Observemos que nos muestra una lista de las versiones que podemos instalar, así como el comando que debemos ejecutar para hacer la instalación.

```
fciencias@fciencias-VirtualBox:~$ java -version
No se ha encontrado la orden «java», pero se puede instalar con:
sudo apt install default-jre          # version 2:1.11-72build2, or
sudo apt install openjdk-11-jre-headless  # version 11.0.19+7~us1-0ubuntu1~22.1
.0.1
sudo apt install openjdk-18-jre-headless # version 18.0.1+10-1
sudo apt install openjdk-17-jre-headless # version 17.0.7+7~us1-0ubuntu1~22.10
.2
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-0ubuntu3~22.10
sudo apt install openjdk-20-jre-headless # version 20.0.1+9~us1-0ubuntu1~22.10
sudo apt install openjdk-8-jre-headless  # version 8u372-ga~us1-0ubuntu1~22.10
fciencias@fciencias-VirtualBox:~$
```

Lo siguiente será ejecutar el siguiente comando para empezar con la instalación de Java 17.

```
$ sudo apt install openjdk-17-jre-headless
```



Nos pedirá nuestra contraseña y nos aparecerá un mensaje preguntando

Se utilizaran 193 MB de espacio de disco adicional después de esta operación.  
¿Desea continuar? [S/n]

Ponemos la opción **S** y damos **Enter** para continuar con la instalación.

Verificamos si se hizo la instalación ejecutando el siguiente comando para revisar la versión de Java.

```
$ java -version
```

Y como podemos observar, el paquete **openjdk** de Java se instaló de forma correcta.

```
fciencias@fciencias-VirtualBox:~$ java -version
openjdk version "17.0.7" 2023-04-18
OpenJDK Runtime Environment (build 17.0.7+7-Ubuntu-0ubuntu122.10.2)
OpenJDK 64-Bit Server VM (build 17.0.7+7-Ubuntu-0ubuntu122.10.2, mixed mode, sharing)
fciencias@fciencias-VirtualBox:~$
```

**Nota:** Hasta ahora se ha instalado Java, sin embargo, aún falta instalar el compilador para que puedas ejecutar tus programas.

## 1.2. Instalación en Fedora

En la mayoría de distribuciones de **Fedora** ya viene instalado el paquete de **Java**, tal es el caso que presentamos a continuación.

El equipo cuenta con la versión de **Fedora 37**. Ejecutamos en la terminal la siguiente línea de comandos para verificar la versión del sistema operativo:

```
$ lsb_release -a
```

La salida que nos da es la siguiente:

```
[miriam@fedora ~]$ lsb_release -a
LSB Version:    :core-4.1-amd64:core-4.1-noarch
Distributor ID: Fedora
Description:    Fedora release 37 (Thirty Seven)
Release:        37
Codename:       ThirtySeven
[miriam@fedora ~]$
```

Ahora, para saber si tenemos instalado el paquete de Java, ejecutamos la siguiente línea de comandos en la consola:

```
$ java --version
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux

Autor: Miriam Torres Bucio

La salida es la siguiente:

```
[miriam@fedora ~]$ java --version
openjdk 17.0.4.1 2022-08-12
OpenJDK Runtime Environment (Red Hat-17.0.4.1.1-3.fc37) (build 17.0.4.1+1)
OpenJDK 64-Bit Server VM (Red Hat-17.0.4.1.1-3.fc37) (build 17.0.4.1+1, mixed mode,
sharing)
[miriam@fedora ~]$ █
```

Lo que quiere decir que contamos con la versión [Java 17](#).

Ahora bien, si queremos instalar la versión [11](#) por ejemplo, debemos ejecutar el siguiente comando en la terminal:

```
$ sudo dnf install -y java-11-openjdk
```

**Nota:** A partir de la versión de [Fedora 35](#) se sustituye el comando '[yum](#)' por el comando '[dnf](#)' para las actualizaciones e instalaciones.

Al terminar con la instalación, nos aparecerá lo siguiente en la terminal:

```
Actualizado:
tzdata-java-2023c-1.fc37.noarch
Instalado:
java-11-openjdk-1:11.0.19.0.7-1.fc37.x86_64
java-11-openjdk-headless-1:11.0.19.0.7-1.fc37.x86_64
mkfontscale-1.2.2-2.fc37.x86_64
ttmkfdir-3.0.9-66.fc37.x86_64
xorg-x11-fonts-Type1-7.5-34.fc37.noarch

¡Listo!
[miriam@fedora ~]$ █
```

Ya tenemos instalado [Java 11](#), sin embargo, aún no lo podemos usar pues al programar se estará usando la versión que tenía previamente instalada. Para cambiarlo, hay que editar las [variables de entorno](#).

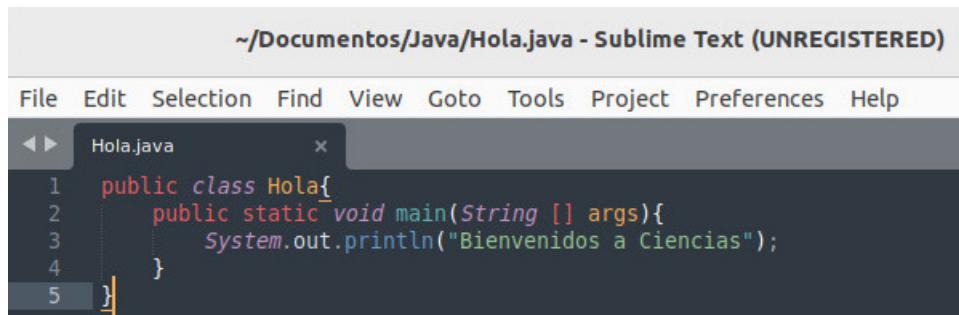
**Nota:** Asegúrate de tener instalado el compilador de Java para poder ejecutar y probar tus programas.



## 2. Instalando el compilador de Java

Para poder ejecutar un programa escrito en Java hacemos uso de `javac` que es el **compilador** de éste lenguaje de programación y su principal objetivo es decirnos si hay errores de **sintaxis** en nuestro programa.

Abrimos nuestro **editor de textos** y escribimos el siguiente código en Java.



```
~/Documentos/Java/Hola.java - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

▶ Hola.java ×
1 public class Hola{
2     public static void main(String [] args){
3         System.out.println("Bienvenidos a Ciencias");
4     }
5 }
```

Observemos que el archivo está guardado dentro del directorio `/Documentos/Java` y el archivo lo nombramos como `Hola.java`.

Para poder **compilar** nuestro programa haremos uso de la terminal de nuestro sistema. Abrimos la consola, Ctrl+Alt+T y ejecutamos los siguientes comandos:

```
$ cd Documentos/Java
$ javac Hola.java
```

Hasta el momento, solo hemos hecho la instalación del lenguaje Java, sin embargo, no hemos instalado su máquina de ejecución que es precisamente su **compilador**. Por lo tanto, al ejecutar las líneas anteriores nos saldrá la siguiente salida:

```
fciencias@fciencias-VirtualBox:~$ cd Documentos/Java
fciencias@fciencias-VirtualBox:~/Documentos/Java$ javac Hola.java
No se ha encontrado la orden «javac», pero se puede instalar con:
sudo apt install default-jdk          # version 2:1.11-72build2, or
sudo apt install openjdk-11-jdk-headless # version 11.0.19+7~us1-0ubuntu1~22.1
0.1
sudo apt install ecj                  # version 3.16.0-1
sudo apt install openjdk-18-jdk-headless # version 18.0.1+10-1
sudo apt install openjdk-17-jdk-headless # version 17.0.7+7~us1-0ubuntu1~22.10
.2
sudo apt install openjdk-19-jdk-headless # version 19.0.2+7-0ubuntu3~22.10
sudo apt install openjdk-20-jdk-headless # version 20.0.1+9~us1-0ubuntu1~22.10
sudo apt install openjdk-8-jdk-headless # version 8u372-ga~us1-0ubuntu1~22.10
fciencias@fciencias-VirtualBox:~/Documentos/Java$
```

Como tenemos instalada la versión 17 de Java, debemos instalar el compilador de dicha versión. Entonces, si instalaste Java 11 debes instalar javac 11.

Ejecutamos el comando que nos da en la terminal y nos pedirá nuestra contraseña:

```
$ sudo apt install openjdk-17-jdk-headless
[sudo contraseña]:
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Al finalizar la instalación, volvemos a compilar nuestro programa [Hola.java](#) y nos compila de manera exitosa.

```
fciencias@fciencias-VirtualBox:~/Documentos/Java$ javac Hola.java
fciencias@fciencias-VirtualBox:~/Documentos/Java$ java Hola
Bienvenidos a Ciencias
fciencias@fciencias-VirtualBox:~/Documentos/Java$ █
```

Ahora si, estás listo para poder ejecutar programas en Java.

### 3. Variables de entorno de Java

Las variables de entorno son un valor dinámico cargado en la memoria, que puede ser utilizado por varios procesos que funcionan de manera simultánea.

En Java se suelen configurar 2 variables de entorno:

- **path:** Es una variable de entorno que informa al Sistema Operativo sobre la ruta de distintos directorios fundamentales para el funcionamiento de los programas.

En la variable PATH debemos indicar donde se encuentran los programas ejecutables de Java necesarios para el desarrollo de aplicaciones como pueden ser el compilador (javac) y el intérprete(java).

Es donde el intérprete de comandos buscará los comandos de ejecución que escribamos en la consola, siempre y cuando no usemos una ruta específica para llamar al comando.

- **java\_home:** Es una variable de entorno que informa al sistema operativo sobre la ruta donde se encuentra instalado Java. Por ejemplo si tenemos instalada más una versión de java, sirve para indicar cuál es la activa en el sistema.

Ahora, procedemos a configurar la variable de entorno.

Determinaremos la ruta de instalación de [Java 11](#), ejecutamos el siguiente comando:

```
$ sudo update-alternatives --config java
```

Nos aparecerá lo siguiente en pantalla. Pulsamos [Enter](#).

```
fciencias@fciencias-VirtualBox:~$ sudo update-alternatives --config java
[sudo] contraseña para fciencias:
Existe 1 opción para la alternativa java (que provee /usr/bin/java).

  Selección    Ruta                      Prioridad  Estado
  * 0            /usr/lib/jvm/java-17-openjdk-amd64/bin/java  1711      modo automático
    1            /usr/lib/jvm/java-17-openjdk-amd64/bin/java  1711      modo manual

Pulse <Intro> para mantener el valor por omisión [*] o pulse un número de selección:
```

Ahora que ya tenemos la ruta de instalación debemos abrir el directorio [/etc/environment](#) y agregar dicha ruta. Para esto, ejecutamos el siguiente comando:

```
$ sudo nano /etc/environment
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

En nuestro caso, solo nos aparece lo siguiente en el archivo:

```
GNU nano 6.4                               /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/">>
```

Editamos el archivo y agregamos la ruta del modo automático, es decir

```
/usr/lib/jvm/java-17-openjdk-amd64/bin/java
```

Lo recomendable es poner un comentario para saber que esa línea la agregamos nosotros.

```
GNU nano 6.4                               /etc/environment *
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/">>
##Agregamos variable de entorno
JAVA_HOME="/usr/lib/jvm/java-17-openjdk-amd64/bin/java"
```

Guardamos los cambios pulsando Ctrl+O, damos Enter y finalmente salimos con Ctrl+X.

Ahora, lo que acabamos de hacer permite que todos los usuarios en el sistema tengan acceso a la variable de entorno JAVA\_HOME, lo siguiente será aplicar los cambios que hemos hecho. Ejecutamos:

```
$ source /etc/environment
```

Y para finalizar y verificar que la variable de entorno que creamos, JAVA\_HOME, haya sido añadida correctamente. Ejecutamos:

```
$ echo $JAVA_HOME
```

La ruta es la que nos debe salir en consola.

```
fciencias@fciencias-VirtualBox:~$ echo $JAVA_HOME
/usr/lib/jvm/java-17-openjdk-amd64/bin/java
fciencias@fciencias-VirtualBox:~$
```

Listo, hemos creado una variable de entorno en Java.



## 4. ¿Cómo instalar un editor de textos?

Una de las herramientas que debes tener instaladas para empezar a programar es un [editor de textos](#) en donde te permita escribir de manera limpia tu código y te permita visualizar errores que puedas tener en la elaboración de un código en java.

En este apartado, veremos dos opciones que puedes considerar a instalar y que te serán de ayuda a lo largo de la carrera.

### 4.1. Sublime Text 3

Este [editor de textos](#)<sup>1</sup> es básico para empezar a programar y el usuario puede cambiar los colores de acuerdo a su preferencia.

**Nota:** Instalamos la versión [Sublime Text 3](#) pues aún es gratuita a diferencia de la versión 4 que hay que pagar por ella.

Sublime Text es recomendado por su simplicidad para empezar a escribir código pues es una interfaz muy sencilla en la que literalmente solo escribimos código.

#### 4.1.1. Instalación en Ubuntu

Para su instalación, vamos a abrir nuestra terminal Ctrl+Alt+T y dicha instalación se hizo en la versión 22.04 de Ubuntu. Lo primero que debemos hacer es asegurarnos de que los paquetes que ya tenemos previamente instalados estén actualizados en nuestra computadora.

```
$ sudo apt update && sudo apt upgrade
```

Lo siguiente será instalar el paquete [curl](#) para poder hacer la instalación.

```
$ sudo apt install curl
```

Nos aparecerá un mensaje que nos dirá:

```
Se utilizarán 193 MB de espacio de disco adicional después de esta operación.  
¿Desea continuar? [S/n]
```

Ponemos la opción [S](#) y damos [Enter](#) para continuar con la instalación.

Se hace la instalación. Despues, debemos [importar](#) la llave del editor de textos ejecutando el siguiente comando:

```
$ curl -fsSL https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add
```

Nos tiene que aparecer lo siguiente en pantalla:

```
fciencias@fciencias-VirtualBox:~$ curl -fsSL https://download.sublimetext.com/s  
ublimehq-pub.gpg | sudo apt-key add  
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).  
OK  
fciencias@fciencias-VirtualBox:~$
```

<sup>1</sup>Hameed, B. S. (n.d.). How to Install Sublime Text 3 on Ubuntu 22.04. Retrieved June 13, 2023, from <https://linuxhint.com/install-sublime-text3-ubuntu-22-04/>



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Lo siguiente será importar el repositorio del editor de textos en nuestro sistema, ejecutamos el siguiente comando:

```
$ sudo add-apt-repository "deb https://download.sublimetext.com/ apt/stable/"
```

Al ejecutar el comando nos saldrá el siguiente mensaje:

```
Añadiendo repositorio.  
Presiona [Enter] para continuar o Ctrl+C para cancelar
```

Presionamos [Enter](#) para continuar con el proceso.

Para finalizar, ejecutamos el siguiente comando para instalar el editor de textos Sublime Text 3.

```
$ sudo apt install sublime-text
```

Al finalizar con la instalación, nos aparecerá lo siguiente en pantalla:

```
fciencias@fciencias-VirtualBox:~$ sudo apt install sublime-text  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
Se instalarán los siguientes paquetes NUEVOS:  
  sublime-text  
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 212 no actualizados.  
Se necesita descargar 16.6 MB de archivos.  
Se utilizarán 52.3 MB de espacio de disco adicional después de esta operación.  
Des:1 https://download.sublimetext.com apt/stable/ sublime-text 4143 [16.6 MB]  
Descargados 16.6 MB en 4s (3 929 kB/s)  
Seleccionando el paquete sublime-text previamente no seleccionado.  
(Leyendo la base de datos ... 212650 ficheros o directorios instalados actualme  
nte.)  
Preparando para desempaquetar ....sublime-text_4143_amd64.deb ...  
Desempaquetando sublime-text (4143) ...  
Configurando sublime-text (4143) ...  
Procesando disparadores para mailcap (3.70+nmu1ubuntu1) ...  
Procesando disparadores para desktop-file-utils (0.26-1ubuntu4) ...  
Procesando disparadores para hicolor-icon-theme (0.17-2) ...  
Procesando disparadores para gnome-menus (3.36.0-1ubuntu3) ...  
fciencias@fciencias-VirtualBox:~$
```

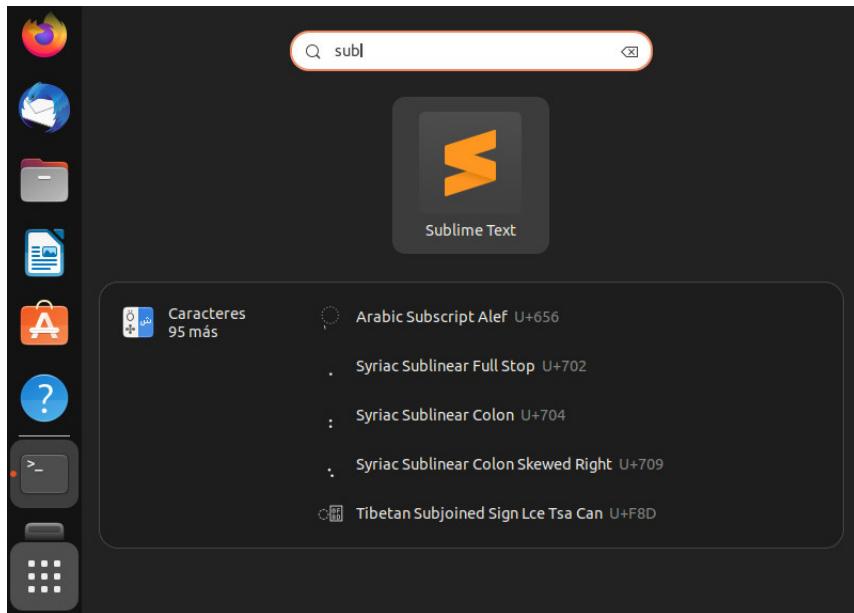


# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Ahora, si nos vamos al buscador de [Ubuntu](#) y ponemos [Sublime](#) podemos ver que está instalado y listo para usarse.



### 4.1.2. Instalación en Fedora

Para la [instalación](#)<sup>2</sup> en [Fedora](#), de la misma manera, haremos uso de la terminal. Primero, vamos a instalar la clave gpg. Para esto, ejecutamos el siguiente comando:

```
$ sudo rpm -v --import https://download.sublimetext.com/sublimehq-rpm-pub.gpg
```

Después seleccionamos la versión a utilizar. En este caso, elegimos la versión estable. Existe la versión en desarrollo, sin embargo, si usamos esa versión podemos tener algunos problemas.

Ejecutamos:

```
$ sudo dnf config-manager --add-repo https://download.sublimetext.com/rpm/stable/x86_64/sublime-text.repo
```

Como último paso, debemos actualizar los repositorios y hacer la instalación del editor.

```
$ sudo dnf install sublime-text
```

Durante la instalación, nos aparecerá el siguiente mensaje:

```
Instalar 1 paquete
```

```
Tamaño total de la descarga: 20 M
Tamaño instalado: 50 M
¿Está de acuerdo [s/N]?:
```

<sup>2</sup>Merino, F. (2017, July 26). Instalar Sublime Text 3 mediante repositorio oficial en Fedora – linuxitos. Retrieved June 14, 2023, from <https://linuxitos.com/main/instalar-sublime-text-3-mediante-repositorio-oficial-en-fedora/>



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Damos la opción de **S**, pulsamos **Enter** y continuamos con la instalación.

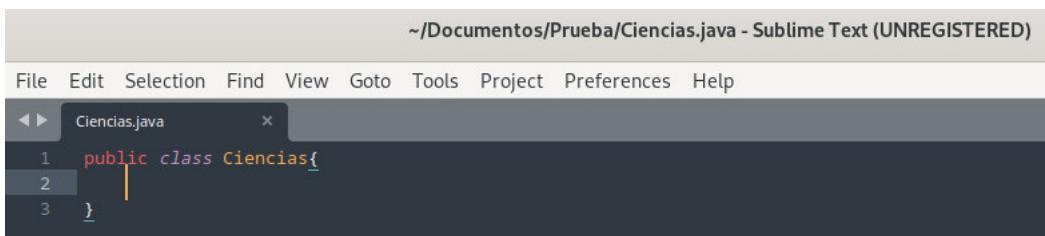
Al finalizar la instalación, nos aparecerá lo siguiente en pantalla:

```
Total 3.6 MB/s | 20 MB 00:05
Ejecutando verificación de operación
Verificación de operación exitosa.
Ejecutando prueba de operaciones
Prueba de operación exitosa.
Ejecutando operación
Preparando : 1/1
Instalando : sublime-text-4143-1.x86_64 1/1
Ejecutando scriptlet: sublime-text-4143-1.x86_64 1/1
Verificando : sublime-text-4143-1.x86_64 1/1

Instalado:
sublime-text-4143-1.x86_64

¡Listo!
[miriam@fedora ~]$ █
```

Y listo, ya tenemos instalado Sublime Text 3 en Fedora. Podemos ir al buscador, abrir el editor y empezar a escribir código.



## 4.2. Visual Studio Code

Visual Studio Code<sup>3</sup> es un editor de textos que está disponible de forma gratuita para la mayoría de los sistemas operativos modernos, como Windows, Linux macOS.

Este editor de textos tiene muchas más funciones que Sublime Text, pero hay que saberlas usar. Es recomendable que la instales si ya tienes un poco más de experiencia con un editor y no se te dificulte usarlo.

Tiene las siguientes características:

- Instala extensiones para agregar nuevos idiomas, temas, depuradores y para conectarse a servicios adicionales. Las extensiones se ejecutan en procesos separados, lo que garantiza que no ralentizarán el editor.
- Tiene código de depuración directamente desde el editor. Hay que iniciar desde las aplicaciones en ejecución y depurar con puntos de interrupción, pilas de llamadas y una consola interactiva.
- Tiene integrada una función de autocompletar que proporciona terminaciones inteligentes basadas en tipos de variables, definiciones de funciones y módulos importados.
- Se puede trabajar con GitHub y otros proveedores.

<sup>3</sup>Noviello. (2022). Cómo instalar Visual Studio Code en Ubuntu 22.04. Noviello.it. <https://noviello.it/es/como-instalar-visual-studio-code-en-ubuntu-22-04/>



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

### 4.2.1. Instalación en Ubuntu

La siguiente instalación la haremos usando el paquete [snapd](#).

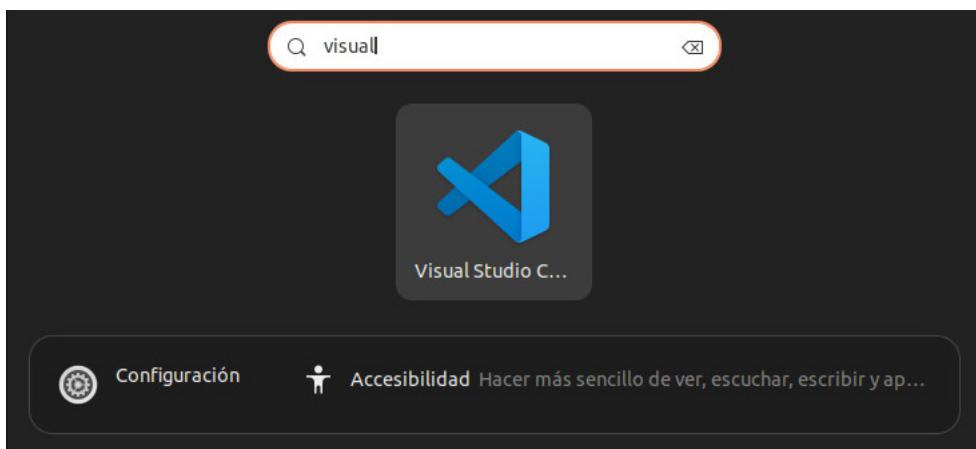
Abrimos nuestra terminal Ctrl+Alt+T y ejecutamos el siguiente comando en la consola:

```
$ sudo snap install code --classic
```

Nos aparecerá lo siguiente en pantalla:

```
fciencias@fciencias-VirtualBox:~$ sudo snap install code --classic
Se ha instalado code 4cb974a7 por Visual Studio Code (vscode✓)
fciencias@fciencias-VirtualBox:~$
```

Y listo, la instalación se hizo de forma rápida. Si nos vamos al buscador y escribimos [Visual](#) nos aparecerá el editor de textos.





# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

### 4.2.2. Instalación en Fedora

Para hacer la instalación en Fedora<sup>4</sup> ejecutaremos los siguientes comandos en la terminal:

```
$ sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
$ sudo sh -c 'echo -e "[code]\nname=Visual Studio Code\nbaseurl=https://
packages.microsoft.com/yumrepos/vscode\nenabled=1\npgpcheck=1\npgpkey=https://
packages.microsoft.com/keys/microsoft.asc" > /etc/yum.repos.d/vscode.repo'
```

Después, verificamos los paquetes y por último hacemos la instalación, esto para versiones actuales de [Fedora](#), para versiones antiguas (Fedora 22 hacia abajo) usamos [yum](#) en lugar de [dnf](#).

```
$ dnf check-update
$ sudo dnf install code
```

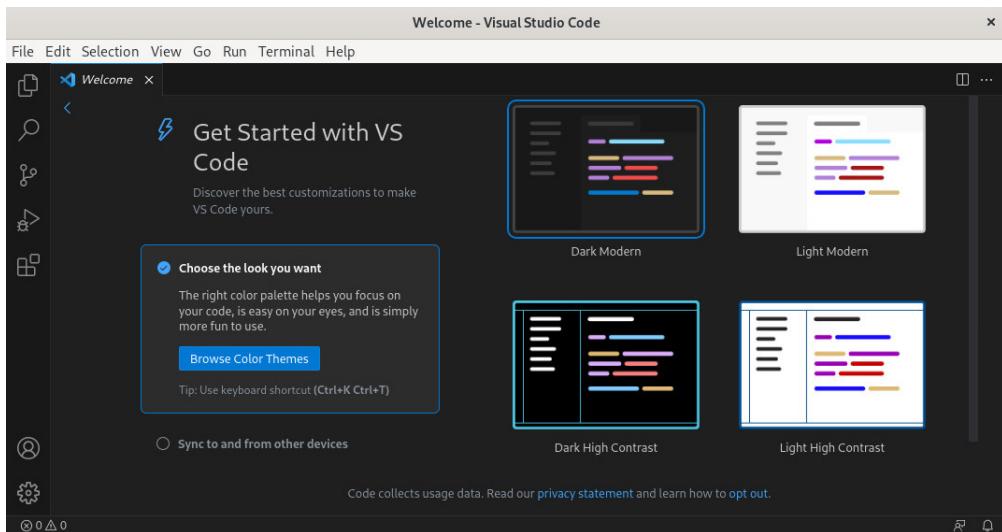
Al finalizar la instalación del editor, nos aparecerá la siguiente información.

```
Total                                         5.6 MB/s | 129 MB   00:23
Ejecutando verificación de operación
Verificación de operación exitosa.
Ejecutando prueba de operaciones
Prueba de operación exitosa.
Ejecutando operación
  Preparando      :
  Instalando     : code-1.79.1-1686587726.el7.x86_64          1/1
  Ejecutando scriptlet: code-1.79.1-1686587726.el7.x86_64          1/1
  Verificando    : code-1.79.1-1686587726.el7.x86_64          1/1

Instalado:
  code-1.79.1-1686587726.el7.x86_64

¡Listo!
[miriam@fedora ~]$
```

Finalmente, si buscamos en las herramientas ya tendremos el editor descargado y listo para usarlo.



<sup>4</sup>Running Visual Studio Code on Linux. (2021, November 3). Retrieved June 14, 2023, from <https://code.visualstudio.com/docs/setup/linux>



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

### 4.3. Emacs

Emacs<sup>5</sup> es un editor de código clásico de GNU, la organización que se encarga de dar soporte al Kernel de Linux con todo un ecosistema de software libre para completar el sistema operativo. Además, permite hacer uso de diversas ayudas para programadores que lo sitúan en un rango profesional.

#### 4.3.1. Instalación en Ubuntu

Para la instalación<sup>6</sup> en Ubuntu solo debemos ejecutar tres comandos en la terminal.  
Empezamos por actualizar los paquetes y repositorios.

```
$ sudo apt update
```

Y empezamos con la instalación directa del editor Emacs.

```
$ sudo apt install emacs
```

En el proceso de instalación nos aparecerá un mensaje:

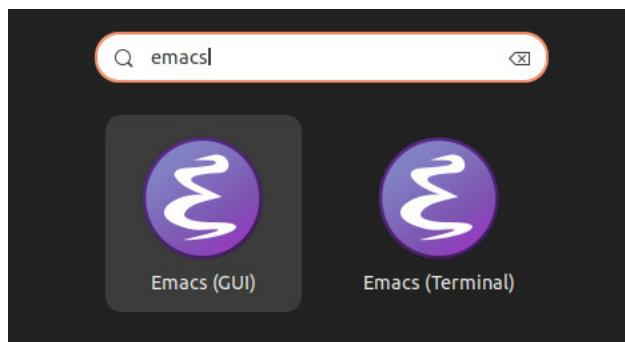
```
Se necesita descargar 36.4 MB de archivos.  
Se utilizarán 117 MB de espacio de disco adicional después de esta operación.  
¿Desea continuar? [S/n]
```

Insertamos la opción S y damos Enter para continuar con la instalación.

Al finalizar la instalación nos aparecerá lo siguiente en pantalla:

```
Install emacsclient-common for emacs
emacsclient-common: Handling install of emacsclient flavor emacs
Install dictionaries-common for emacs
install/dictionaries-common: Byte-compiling for emacsclient flavour emacs
Configurando emacs (1:27.1+1-3ubuntu5) ...
Procesando disparadores para hicolor-icon-theme (0.17-2) ...
Procesando disparadores para gnome-menus (3.36.0-1ubuntu3) ...
Procesando disparadores para libc-bin (2.36-0ubuntu4) ...
Procesando disparadores para man-db (2.10.2-2) ...
Procesando disparadores para install-info (6.8-6) ...
Procesando disparadores para mailcap (3.70+nmu1ubuntu1) ...
Procesando disparadores para desktop-file-utils (0.26-1ubuntu4) ...
fciencias@fciencias-VirtualBox:~$
```

Finalmente, si lo buscamos en las herramientas nos aparecerá y estará disponible para usar.



<sup>5</sup>Snapcraft. (2023, June 4). Snapcraft. Retrieved June 14, 2023, from <https://snapcraft.io/install/emacs/fedora#install>

<sup>6</sup>Maurya, H. (2022). 3 ways to install Emacs text editor on Ubuntu 20.04. Linux Shout. <https://linux.how2shout.com/3-ways-to-install-emacs-text-editor-on-ubuntu-20-04/>



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

### 4.3.2. Instalación en Fedora

Para hacer la instalación, haremos uso del paquete [snapd](#). Abrimos la terminal y ejecutamos los siguientes comandos:

```
$ sudo dnf install snapd
```

En el proceso de descarga nos aparecerá un mensaje que dice:

```
Tamaño total de la descarga: 25 M
¿Está de acuerdo [s/N]?:
```

Ponemos la opción de [S](#) y damos [Enter](#) para que siga con la instalación. Finalizada la instalación del paquete, nos aparecerá la siguiente información:

```
Actualizado:
  selinux-policy-37.21-2.fc37.noarch
  selinux-policy-targeted-37.21-2.fc37.noarch
Instalado:
  snap-confine-2.58.3-1.fc37.x86_64           snapd-2.58.3-1.fc37.x86_64
  snapd-selinux-2.58.3-1.fc37.noarch

¡Listo!
[miriam@fedora ~]$
```

Después de haber hecho la instalación, es importante reiniciar el equipo. Ejecutamos el siguiente comando:

```
$ reboot
```

Al iniciar de nuevo la computadora, ejecutamos el siguiente comando el cuál solo nos pedirá nuestra contraseña.

```
$ sudo ln -s /var/lib/snapd/snap /snap
[sudo] contraseña:
```

Ahora, instalamos el editor.

```
$ sudo snap install emacs --classic
```

Ya instalado el editor, nos aparecerá la siguiente información:

```
[miriam@fedora ~]$ sudo ln -s /var/lib/snapd/snap /snap
[sudo] contraseña para miriam:
[miriam@fedora ~]$ sudo snap install emacs --classic
2023-06-14T11:48:59-05:00 INFO Waiting for automatic snapd restart...
emacs 28.2 from Alex Murray (alexmurray*) installed
[miriam@fedora ~]$
```



# Facultad de Ciencias

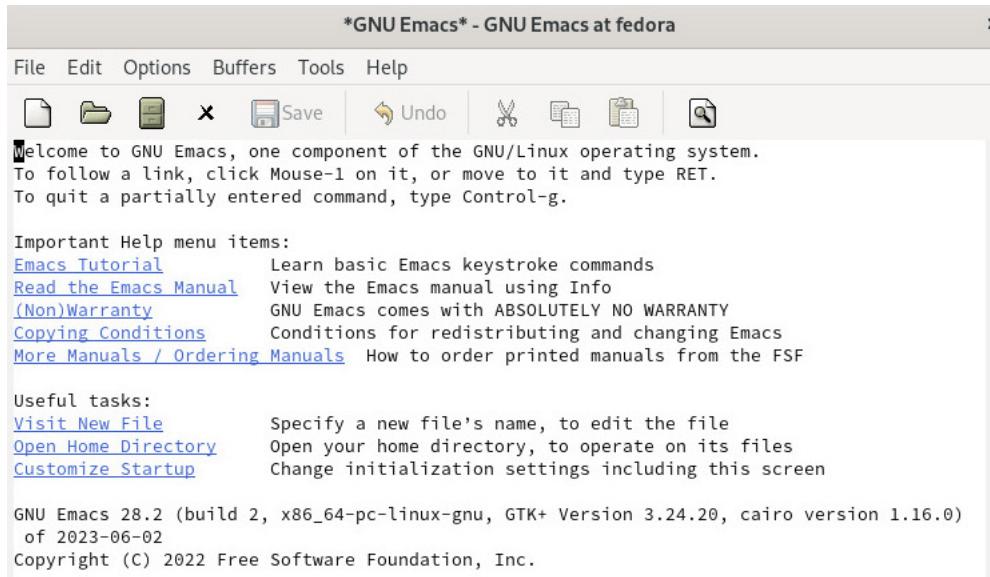
## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Podemos ejecutar en consola

```
$ emacs
```

o simplemente buscarlo en las herramientas y ya tendremos el editor.



## 4.4. Vim

El problema de aprender **Vim**<sup>7</sup> no es que sea difícil, es que hay que seguir aprendiéndolo siempre. Algunas de las características con las que cuenta son las siguientes:

- **Corrección ortográfica:** Se activa por medio del comando `set spell`.  
Los diccionarios para cada una de las lenguas pueden descargarse y ampliarse manualmente.
- **Función autocompletar:** Se puede utilizar en el modo insertar mediante las combinaciones de teclas `Ctrl+N` o `Ctrl+P`.
- **Función hacer/deshacer ilimitada:** Los cambios pueden deshacerse o restablecerse de manera ilimitada, incluso si se ha cerrado el editor Linux.

### 4.4.1. Instalación en Ubuntu

La instalación es sencilla.

Primero debemos actualizar los paquetes y repositorios.

```
$ sudo apt-get update
```

Y después hacemos la instalación:

```
$ sudo apt-get install vim
```

<sup>7</sup>Equipo editorial de IONOS. (2023). Vim, un editor de textos basado en Linux con una amplia funcionalidad. IONOS Digital Guide. <https://www.ionos.mx/digitalguide/servidores/herramientas/editores-linux-como-editar-codigo-con-vim/>



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

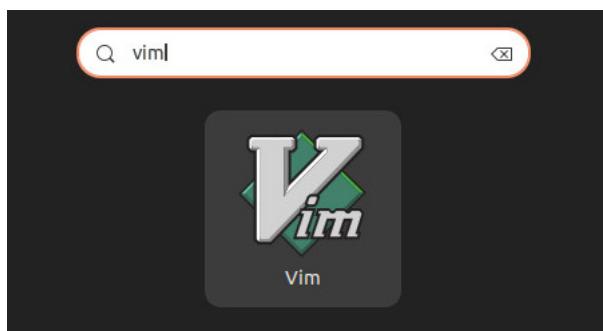
Durante la instalación nos aparecerá el siguiente mensaje:

```
Se utilizarán 40.6 MB de espacio de disco adicional después de esta operación.  
¿Desea continuar? [S/n]:
```

Damos la opción **S**, presionamos **Enter** y continuamos con la instalación.  
Al finalizar la instalación, nos aparecerá lo siguiente en pantalla:

```
Configurando vim-tiny (2:9.0.0242-1ubuntu1.4) ...  
Procesando disparadores para mailcap (3.70+nmu1ubuntu1) ...  
Procesando disparadores para desktop-file-utils (0.26-1ubuntu4) ...  
Procesando disparadores para hicolor-icon-theme (0.17-2) ...  
Procesando disparadores para gnome-menus (3.36.0-1ubuntu3) ...  
Procesando disparadores para man-db (2.10.2-2) ...  
fciencias@fciencias-VirtualBox:~$ █
```

Finalmente, si lo buscamos en nuestras herramientas lo encontraremos y podremos empezar a usarlo.



### 4.4.2. Instalación en Fedora

En Fedora, la instalación es similar a la de Ubuntu, basta con ejecutar dos comandos.  
Primero, verificamos si están actualizados los paquetes que tenemos instalados.

```
$ sudo dnf update
```

Recuerda que **dnf** se utiliza para versiones recientes de Fedora, si cuentas con una versión más antigua, debes usar **yum**.

Después de haber hecho la actualización, hacemos la instalación usando el siguiente comando:

```
$ sudo dnf install vim-enhanced
```

Durante la instalación, nos aparecerá el siguiente mensaje:

```
Instalar      6 Paquetes  
Actualizar    2 Paquetes  
  
Tamaño total: 10 M  
Tamaño total de la descarga: 9.7 M  
¿Está de acuerdo [s/N]?:
```

Damos la opción **S**, presionamos **Enter** y continuamos con la instalación.



# Facultad de Ciencias

---

## UNAM

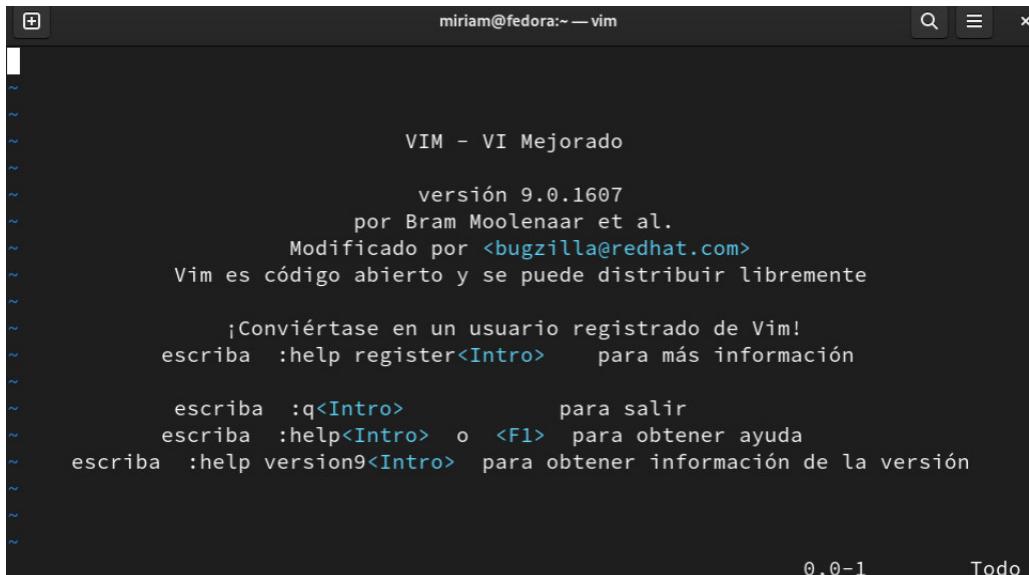
Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

Al finalizar la instalación, nos dirá lo siguiente en pantalla:

```
Actualizado:
  vim-data-2:9.0.1607-1.fc37.noarch      vim-minimal-2:9.0.1607-1.fc37.x86_64
Instalado:
  gpm-libs-1.20.7-41.fc37.x86_64          libsodium-1.0.18-10.fc37.x86_64
  vim-common-2:9.0.1607-1.fc37.x86_64    vim-enhanced-2:9.0.1607-1.fc37.x86_64
  vim-filesystem-2:9.0.1607-1.fc37.noarch xxd-2:9.0.1607-1.fc37.x86_64

:Listo!
[miriam@fedora ~]$
```

Ya hecha la instalación, podemos poner directamente en la consola `vim` y podemos empezar a usarlo.



The screenshot shows a terminal window titled "miriam@fedora:~ — vim". The window displays the Vim 9.0.1607 welcome message, which includes information about the version, authors, and basic usage instructions. The message ends with "0,0-1 Todo".

```
VIM - VI Mejorado
versión 9.0.1607
por Bram Moolenaar et al.
Modificado por <bugzilla@redhat.com>
Vim es código abierto y se puede distribuir libremente

¡Conviértase en un usuario registrado de Vim!
escriba :help register<Intro> para más información

escriba :q<Intro> para salir
escriba :help<Intro> o <F1> para obtener ayuda
escriba :help version9<Intro> para obtener información de la versión

0,0-1 Todo
```

**Nota:** Para empezar a usar Vim, puedes apoyarte en el [siguiente link](#)<sup>a</sup>, es un pequeño tutorial en donde podrás aprender lo básico de este editor.

<sup>a</sup>Alejandro. (2012, June 5). Usando VIM: Tutorial Básico. Desde Linux. Retrieved June 15, 2023, from <https://blog.desdelinux.net/usando-vim-tutorial-basico/>



## 5. Introducción a GitHub;



[GitHub](#) es una plataforma que ayuda a los programadores a mantener a salvo el código que escribimos. Es de mucha utilidad, permite trabajar en equipo con la gran ventaja de poder subir cambios al código sin la necesidad de 'copiar y pegar' ciertas líneas.

Podemos descargar las actualizaciones que se hayan efectuado, siempre y cuándo éstas se hayan subido al [repositorio](#).

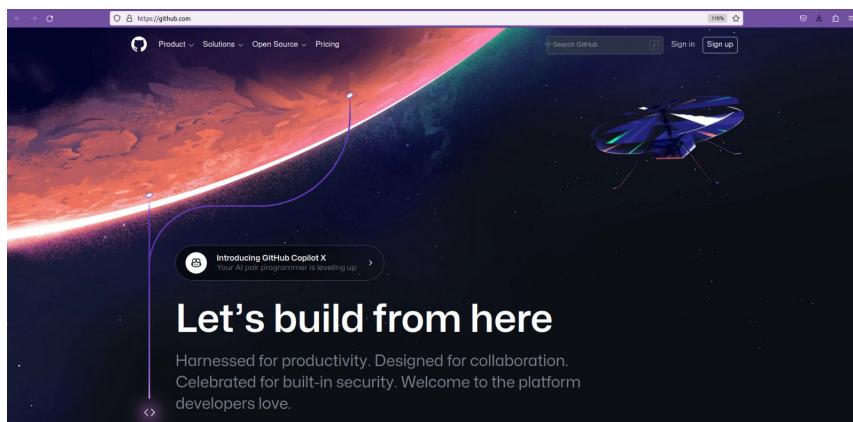
Al entrar a la carrera, es importante que sepas como se usa esta plataforma para que puedas tener tu código guardado y no se pierda ya que estará guardado de forma segura.

En esta sección, se mencionarán los pasos que debes hacer para crear una cuenta en esta plataforma y puedas subir a tu repositorio código que hayas creado. Además se pondrá un ejemplo de cómo es que se sube el código y cómo se descargan los cambios de un repositorio.

### 5.1. ¿Cómo crear una cuenta en GitHub?

Para crear una cuenta en esta plataforma, hay que seguir los siguientes pasos:

- El requisito para poder crear una cuenta en esta plataforma es tener un correo electrónico. Es recomendable que uses el correo de dominio [ciencias](#) que otorga la facultad cuando recién se entra a la carrera.
- Entramos a la página principal de [GitHub](#) la cuál aparece de la siguiente manera:



Iremos a la esquina superior derecha y daremos click en la opción que dice [Sing in](#).



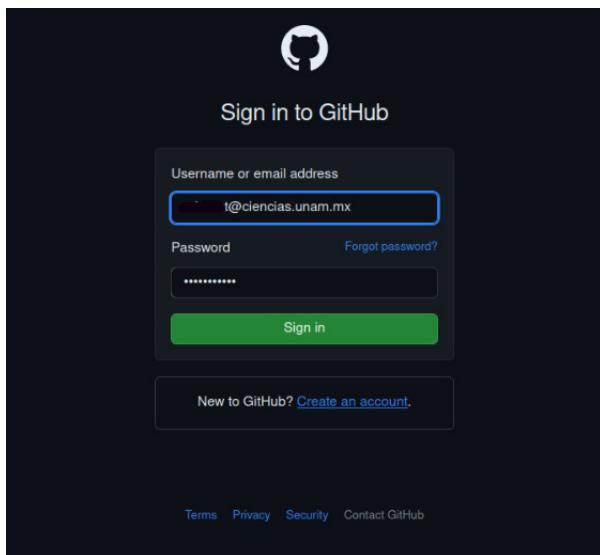
# Facultad de Ciencias

---

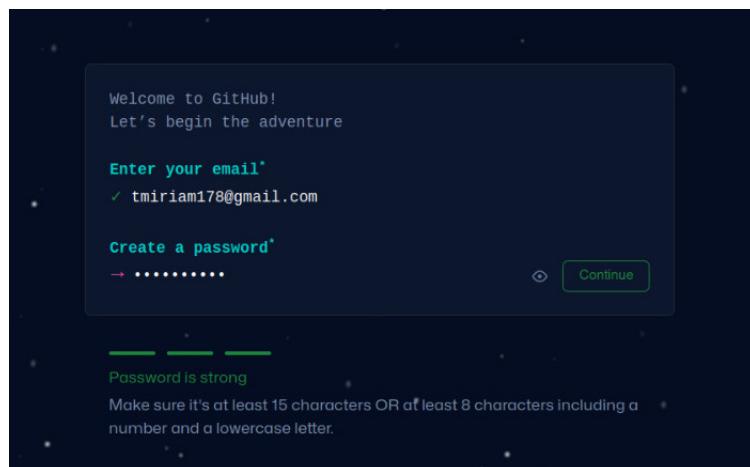
## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Aquí nos dará la opción de ingresar, esto en caso de ya tener una cuenta o de no tenerla, crear una cuenta.  
Damos la opción de [crear una cuenta](#).



- Después, nos redirigirá a una nueva pantalla en donde podremos ingresar el correo electrónico y una contraseña.  
También nos da la opción de que la contraseña sea una generada por la plataforma, ésta se guardará en el navegador que estemos usando y al iniciar sesión no tendremos que ponerla.





# Facultad de Ciencias

---

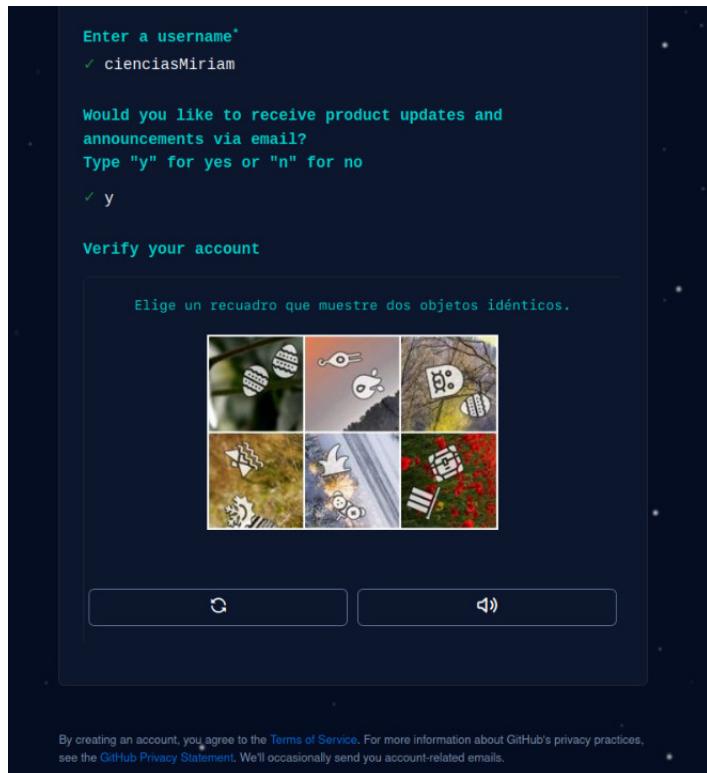
## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Al aceptar la contraseña que hayamos elegido, nos pedirá que ingresemos un **username** que será el nombre que estaremos usando en la plataforma.

También nos preguntará si queremos recibir informes a través del correo electrónico. Podemos decir que no, esto es opcional.

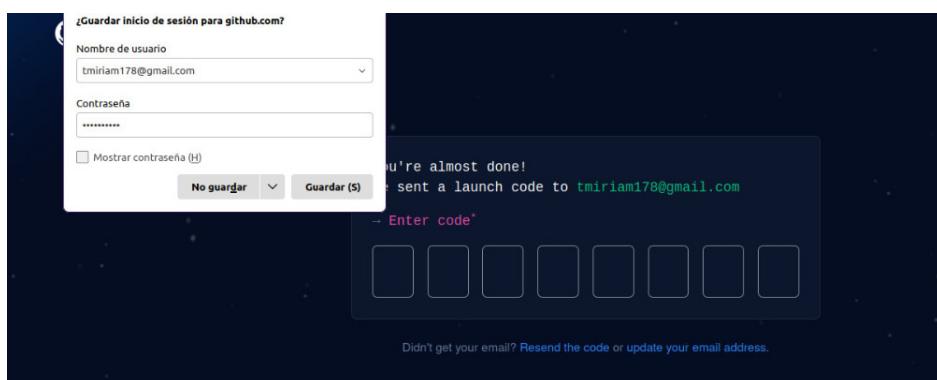
Y finalmente, pasamos una prueba de verificación.



Al finalizar, damos click en **crear cuenta**.

- Después nos aparecerá el siguiente mensaje en donde podremos guardar la contraseña en el navegador para futuros inicios de sesión.

Por otro lado, GitHub nos mandará un código de verificación al correo electrónico con el que creamos la cuenta. Lo insertamos.





# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Después, nos desplegará una lista de opciones que debemos marcar para la utilización del repositorio. Como la cuenta es para empezar a aprender a usar esta plataforma, puedes solo marcar la opción de **Collaborative coding**. Por otro lado, puedes marcar más opciones, sin embargo, algunas solo son gratis por un periodo de tiempo, pasado éste debes pagar por ello.

The tools you need to build what you want.  
Soup to nuts, GitHub has it all.

What specific features are you interested in using?  
Select all that apply so we can point you to the right GitHub plan.

- Collaborative coding  
Codespaces, Pull requests, Notifications, Code review, Code review assignments, Code owners, Draft pull requests, Protected branches, and more.
- Automation and CI/CD  
Actions, Packages, APIs, GitHub Pages, GitHub Marketplace, Webhooks, Hosted runners, Self-hosted runners, Secrets management, and more.
- Security  
Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.
- Client Apps  
GitHub Mobile, GitHub CLI, and GitHub Desktop.
- Project Management  
Projects, Labels, Milestones, Issues, Unified Contribution Graph, Org activity graph, Org dependency insights, Repo insights, Wikis, and GitHub Insights.
- Team Administration  
Organizations, Invitations, Team sync, Custom roles, Domain verification, Audit Log API, Repo creation restriction, and Notification restriction.
- Community  
GitHub Marketplace, GitHub Sponsors, GitHub Skills, and Electron.

Continue

- Al continuar con el proceso, nos saldrá lo siguiente en pantalla:

Welcome to GitHub  
We are glad you're here.

How many team members will be working with you?  
This will help us guide you to the tools that are best suited for your projects.

Just me    2 - 5    5 - 10  
10 - 20    20 - 50    50+

Are you a student or teacher?  
Student    Teacher

Continue

Aquí, debemos decir que la cuenta es personal, solo la usaremos nosotros. En caso de hacer un trabajo en equipo, podemos mandar una invitación, lo veremos en un ejemplo más adelante.

Además, debemos poner la opción de ser estudiantes.

Damos click en Continuar.



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Se nos desplegará una pantalla. Aquí, seleccionamos la opción de [Continue for free](#) para evitar pagar algunos de los servicios que nos ofrece la plataforma.

The screenshot shows the GitHub landing page with a dark background. At the top, it says "Learn to ship software like a pro." and "GitHub gives students free access to the best developer tools so they can learn by doing." Below this, there are two main sections: "Free" and "Get additional student benefits".  
**Free:**

- Unlimited public/private repositories
- 2,000 CI/CD minutes/month (Free for public repositories)
- 500MB of Packages storage (Free for public repositories)
- 120 core-hours of Codespaces compute
- 15GB of Codespaces storage
- Community support

  
**Get additional student benefits:**

- GitHub Pro**
  - Protect your branches
  - Draft pull requests
  - Pages and Wikis
  - 3,000 CI/CD minutes/month (Free for public repositories)
  - 2GB of Packages storage (Free for public repositories)
  - 180 core-hours of Codespaces compute
  - 20GB of Codespaces storage
  - Web-based support
- GitHub Student Developer Pack**
  - Free access to the industry's best developer tools (Hundreds of others, including DigitalOcean, Microsoft Azure, Heroku, MongoDB, Datadog, Hellip, and Stripe)
- GitHub Campus Expert training**
  - Enrich your college technical community (Hundreds of others across the globe help enrich their campus with training, mentorships, and support from GitHub)

At the bottom, there are two buttons: "Continue for free" and "Apply for your GitHub student benefits".

- Finalmente creamos nuestra cuenta en GitHub.

The screenshot shows the GitHub user profile page for "cienciasMiriam". The top navigation bar includes "Pull requests", "Issues", "Codespaces", "Marketplace", and "Explore".  
On the left, there are sections for "Create your first project" (with "Create repository" and "Import repository" buttons), "Recent activity" (with a note about linking to other GitHub actions), and "Tools of the trade".  
The main area shows the user's profile information:

- "For you" tab is selected, showing "Following".
- "Start writing code" button.
- "Start a new repository" section with "Public" and "Private" options, and a "Create a new repository" button.
- "Introduce yourself with a profile README" section, showing a partial README file with items 1-6.
- "Latest changes" sidebar with activity logs (5 minutes ago, 6 minutes ago, 1 hour ago, 2 hours ago).
- "Simplify your development workflow with a GUI" section with a "GitHub Desktop" icon.
- "Install a powerful code editor" section with a "Visual Studio Code" icon.



## 5.2. ¿Cómo crear un repositorio en GitHub?

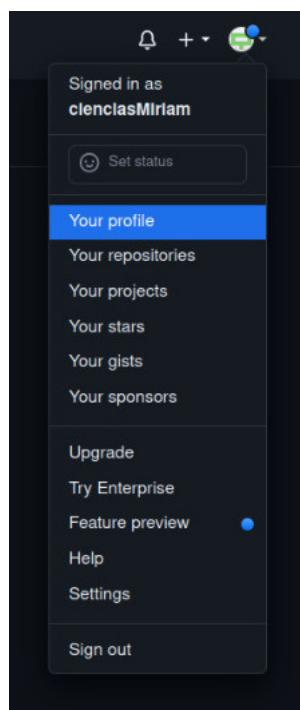
Un [repositorio](#) es en donde tenemos todos los archivos de un proyecto que tengamos elaborado. Podemos verlo como una carpeta que contiene dichos archivos y dicha carpeta la podemos gestionar y controlar las versiones del código elaborado.

Una de las ventajas de usar GitHub es que permite almacenar nuestros trabajos dando así la oportunidad a millones de personas de todo el mundo a cooperar en ellos pues podemos poner nuestro código público.

Para crear un repositorio y subir un proyecto, seguimos los siguientes pasos:

- Al iniciar sesión, debemos ir directamente a nuestro perfil. Para esto debemos ir a la esquina superior derecha y dar click en el ícono donde se encuentra nuestra foto de perfil que nos da por defecto GitHub que, por lo general es de color verde.

Damos click en la opción de [Your profile](#).



**Nota:** La foto de perfil la puedes cambiar por una que tu elijas.



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux

Autor: Miriam Torres Bucio

- Nos tendrá que aparecer la siguiente pantalla.

Aquí está la opción de [Repositories](#)

The screenshot shows a GitHub profile for the user 'cienciasMiriam'. At the top, there are tabs for Overview, Repositories, Projects, Packages, and Stars. The 'Repositories' tab is selected. A large circular profile picture is centered. Below it, the username 'cienciasMiriam' is displayed, along with a 'Edit profile' button and a note that the user joined 1 hour ago. The main content area shows a message: 'You don't have any public repositories yet.' Below this, there is a 'contribution graph' for the last year, which is currently empty except for one small green square in the June column. A tooltip explains what the graph represents: 'This is your contribution graph. Your first square is for joining GitHub and you'll earn more as you make additional contributions. More contributions means a higher contrast square color for that day. Over time, your chart might start looking something like this.' There is also a link to 'Read the Hello World guide'.

- Justo en la sección de [Repositories](#) aparecerán nuestros proyectos que tengamos subidos. Por el momento, como es nueva nuestra cuenta, no tendremos ningún proyecto.

Para crear uno nuevo, ponemos la opción de [New](#).

The screenshot shows the same GitHub profile for 'cienciasMiriam' as the previous one, but with a different message: 'cienciasMiriam doesn't have any public repositories yet.' This indicates that the user has since created a repository. The 'Repositories' tab is still selected, and a prominent green 'New' button is visible at the top right of the main content area. The rest of the interface is identical to the previous screenshot, including the navigation bar, profile picture, and footer links.



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Para la creación de un repositorio, nos pedirá llenar los siguientes campos:

- Repository name:** El nombre de nuestro repositorio o cómo llamaremos a nuestro proyecto. En nuestro caso, lo nombramos [HolaMundo](#).
- Descripción:** Este campo lo podemos dejar vacío o explicar en qué consiste nuestro proyecto. En nuestro caso, pusimos de descripción [Mi primer repositorio](#).
- Público o privado:** En esta opción podemos decidir quién puede ver nuestro proyecto. Evidentemente si lo ponemos público cualquier persona que tenga nuestro link del repositorio puede verlo y si es privado, solo nosotros o un grupo pequeño de personas que hayamos elegido lo podemos ver.
- Inicializamos el repositorio:** Aquí podemos añadir un [README.file](#), esto es opcional. Sin embargo, al añadirlo cambia el proceso de elaboración del repositorio.
- Add .gitignore y Choose a license:** Por el momento dejamos el valor por defecto que tienen estas opciones. La primera, su función es ignorar archivos de las extensiones que le indiquemos, por ejemplo, los archivos .class; y el segundo nos indica las licencias que usamos, por ejemplo, si usamos Eclipse que es un IDE para programar.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (\*).

Owner \* Repository name \*

cienciasMiriam / HolaMundo HolaMundo is available.

Great repository names are short and memorable. Need inspiration? How about [studious-giggle](#) ?

Description (optional)

Mi primer repositorio

**Public** Anyone on the internet can see this repository. You choose who can commit.

**Private** You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

You are creating a public repository in your personal account.

Create repository

Al finalizar, damos click en el botón [Create repository](#).

Y listo, tenemos nuestro primer repositorio creado.



# Facultad de Ciencias

---

## UNAM

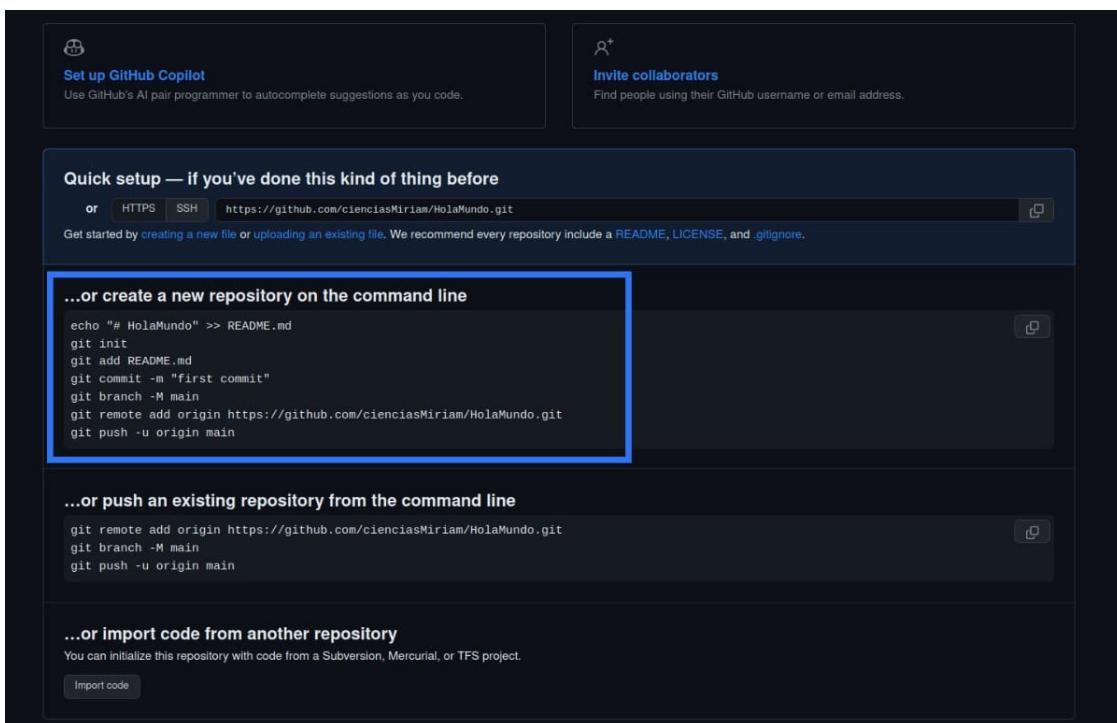
Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

### 5.3. ¿Cómo subir un proyecto a GitHub?

En esta sección veremos como subir un proyecto a un repositorio que acabamos de crear a nuestra cuenta en la plataforma de GitHub.

Seguimos los siguientes pasos:

- En el paso anterior, al crear el repositorio nos debió aparecer la siguiente pantalla con las opciones que se muestran a continuación:



Lo que haremos será ejecutar los comandos que marcamos en el recuadro color azul.

- Vamos a abrir nuestra terminal en nuestra computadora.

Uno de los requisitos para poder usar GitHub es tener instalado el paquete [Git](#) en nuestra computadora. Para saber si lo tenemos instalado, escribimos en la terminal:

```
$ git
```

Si no tenemos instalado git, nos aparecerá lo siguiente:

```
fciencias@fciencias-VirtualBox:~$ git
No se ha encontrado la orden «git», pero se puede instalar con:
sudo apt install git
fciencias@fciencias-VirtualBox:~$ █
```

Ejecutamos el comando que nos indica en la terminal.

```
$ sudo apt install git
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

Durante la instalación, nos aparecerá el siguiente mensaje:

```
fciencias@fciencias-VirtualBox:~$ sudo apt install git
[sudo] contraseña para fciencias:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  git-man liberror-perl
Paquetes sugeridos:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
Se instalarán los siguientes paquetes NUEVOS:
  git git-man liberror-perl
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 85 no actualizados.
Se necesita descargar 4 332 kB de archivos.
Se utilizarán 22.0 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] ■
```

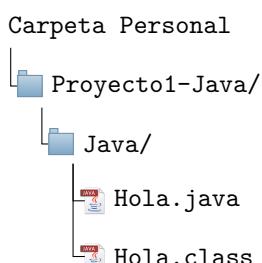
Ponemos la opción **S**, pulsamos **Enter** y seguimos con la instalación.

Ahora, ya teniendo instalado **Git** en nuestra computadora, lo siguiente será ejecutar los comandos que ya mencionamos con anterioridad.

- Escribimos la ruta en donde tengamos nuestro proyecto a subir.

```
$ cd /Proyecto1-Java
```

En un principio, nuestro proyecto está estructurado de la siguiente manera:



Es decir, está dentro del directorio [.../Proyecto1-Java/Java](#) y solo son dos archivos. Podemos deducir que nuestro proyecto se encuentra dentro de la carpeta **Java**.

- Ejecutamos el siguiente comando:

```
$ git init
```

Nos tiene que salir la siguiente información en consola:

```
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ git init
Reinizializado el repositorio Git existente en /home/fciencias/Proyecto
1-Java/.git/
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

**Nota:** El siguiente comando a ejecutar es:

```
$ git add README.md
```

Sin embargo, marcará error pues el comando *add* añade archivos con el nombre que pongamos a continuación y nosotros no tenemos un archivo con nombre README.md

Por otro lado, si añadimos el carácter *{.}* después del comando *add* le estamos diciendo que añada todos los archivos existentes en nuestra carpeta del proyecto que queramos subir a la plataforma.

- Entonces, ejecutamos los siguientes comandos:

```
$ git add .  
$ git commit -m "Proyecto inicializado"
```

El primero añade todos los archivos que tenemos en el repositorio y el segundo pone el comentario que nosotros elijamos; esto es importante pues a través de estos mensajes se puede saber qué cambios se hizo en el proyecto y si se trabaja en equipo todos están enterados de los cambios que se hicieron.

Cuando es nuevo el repositorio y no lo hemos usado suele aparecernos este tipo de mensajes después de hacer un [commit](#):

```
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ git add .  
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ git commit -m "Proyecto inicializado"  
Identidad del autor desconocido  
  
*** Por favor cuéntame quién eres.  
  
Ejecuta  
  
git config --global user.email "you@example.com"  
git config --global user.name "Tu Nombre"  
  
para configurar la identidad por defecto de tu cuenta.  
Omite --global para configurar tu identidad solo en este repositorio.  
  
fatal: no es posible auto-detectar la dirección de correo (se obtuvo 'fciencias@fciencias-VirtualBox.(none)')  
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ █
```

Esto es para verificar nuestra identidad. Por lo tanto, ejecutamos los comandos que nos dice en consola. En nuestro caso, proporcionamos el correo electrónico y nombre de usuario de nuestra cuenta de GitHub:

```
$ git config --global user.email tmiriam178@gmail.com  
$ git config --global user.name cienciasMiriam
```

Dichos comandos no producen ninguna salida en consola.



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

Ahora, volvemos a ejecutar nuestro commit:

```
$ git commit -m "Proyecto inicializado"
```

Y ahora si, nos debe salir los cambios en el repositorio:

```
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ git commit -m "Proyecto inicializado"
[master (commit-raíz) a84ea94] Proyecto inicializado
 2 files changed, 5 insertions(+)
  create mode 100644 Java/Hola.class
  create mode 100644 Java/Hola.java
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ █
```

- Lo siguiente será indicarle a qué **rama** de nuestro repositorio se van a subir los cambios establecidos del proyecto en nuestro repositorio. Por lo general, se trabaja siempre en la rama **main** o **master**, sin embargo, la que da GitHub por defecto es la rama main.

Entonces, ejecutamos el siguiente comando:

```
$ git branch -M main
```

- Subimos nuestro proyecto al repositorio. Ejecutamos el comando con la URL que nos proporcionó GitHub:

```
$ git remote add origin https://github.com/cienciasMiriam/HolaMundo.git
```

**Nota:** Podemos copiar el comando Ctrl+C y para pegar la información en la terminal de Linux, lo haremos con las teclas Ctrl+Shift+V.

- Por último, ejecutamos el siguiente comando, el cuál **empujará** (subirá) los cambios correspondientes en la rama **main**:

```
$ git push -u origin main
```

Ahora bien, puede que al tratar de verificar nuestros datos y éstos sean correctos, es probable que salga el siguiente error:

```
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ git push -u origin main
Username for 'https://github.com': cienciasMiriam
Password for 'https://cienciasMiriam@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Autenticación falló para 'https://github.com/cienciasMiriam/HolaMundo.git'
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ █
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Para solucionar esto, debemos generar un [token](#).

Ya generado nuestro token, volvemos a hacer el [push](#), es decir, ejecutamos nuevamente:

```
$ git push -u origin main
```

Sin embargo, en nuestra contraseña pondremos nuestro token.

**Nota:** Como la estructura del token es una cadena de caracteres de mayúsculas, minúsculas y a veces números, te recomendamos que copies y pegues la contraseña para que sea más fácil ponerla.

La salida en consola debe ser parecida a la siguiente:

```
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$ git push -u origin mai
n
Username for 'https://github.com': cienciasMiriam
Password for 'https://cienciasMiriam@github.com':
Enumerando objetos: 5, listo.
Contando objetos: 100% (5/5), listo.
Compresión delta usando hasta 3 hilos
Comprimiendo objetos: 100% (4/4), listo.
Escribiendo objetos: 100% (5/5), 698 bytes | 349.00 KiB/s, listo.
Total 5 (delta 0), reusados 0 (delta 0), pack-reusados 0
To https://github.com/cienciasMiriam/HolaMundo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
fciencias@fciencias-VirtualBox:~/Proyecto1-Java$
```

Nos da información de los archivos que se insertaron, los que se eliminaron y los reusados. También nos dice en qué rama se subieron nuestros cambios.

- Finalmente, si vamos a nuestro repositorio podemos ver que ya está creado y además, tiene nuestro proyecto, así como nuestro [commit](#) que está marcado en el recuadro color azul.

The screenshot shows a GitHub repository named 'HolaMundo'. The repository details indicate it is public, has 1 branch (main), and 0 tags. The 'main' branch was last updated yesterday with a single commit from user 'cienciasMiriam' with the message 'Proyecto inicializado'. The repository is described as 'Mi primer repositorio'. On the right side, there are statistics: 0 stars, 1 watching, and 0 forks. A button at the bottom right says 'Add a README'.



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Si damos click en nuestro letrero de commit, podremos ver la siguiente información:

- Los archivos de nuestro proyecto, que están enmarcados en el cuadrado color rojo.
- El contenido de nuestros archivos, enmarcado en el recuadro color azul.
- La rama en la que estamos trabajando, en este caso, la rama main enmarcada en el recuadro color morado.
- Si lo deseamos, podemos poner comentarios en nuestro código y publicarlos. Esto se hace en el espacio que está señalado con la flecha color verde.

The screenshot shows a GitHub commit page for a repository named "Proyecto inicializado". The commit was made by "cienciasMiriam" yesterday. It shows 2 changed files with 5 additions and 0 deletions. The commit message is "main". The commit hash is "a84ea94". The commit is part of a single parent. The code editor shows a Java class named "Hola.java" with the following content:

```
public class Hola{    public static void main(String [] args){        System.out.println("Bienvenidos a Ciencias");    }}
```

Y listo, finalmente subimos un proyecto a un repositorio nuevo.



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

### 5.4. ¿Cómo generar un token en GitHub?

Los tokens de acceso personal son una alternativa al uso de contraseñas para la autenticación en GitHub para subir un proyecto por ejemplo, y están destinados a acceder a los recursos de GitHub en su nombre.

Actualmente, GitHub admite dos tipos de tokens de acceso personal: tokens de acceso personal detallados y tokens de acceso personal (clásico). GitHub recomienda usar tokens de acceso personal detallados en lugar de tokens de acceso personal (clásicos) siempre que sea posible.

Para crear un token haremos los siguientes pasos:

- Iremos a nuestro perfil de nuestra cuenta de GitHub y daremos click en nuestra foto de perfil que se encuentra en la esquina superior derecha y entraremos en la opción [Settings](#) tal y como lo muestra la [Figura 1](#).
- Despues nos saldrá una pantalla con toda la configuración de nuestra cuenta. Sin embargo, lo que nos interesa es el listado de opciones que aparecen del lado izquierdo de la pantalla [Figura 2](#) y [Figura 3](#).
- Daremos click en la opción [Developer settings](#), es la última opción del listado y se encuentra en la [Figura 3](#).

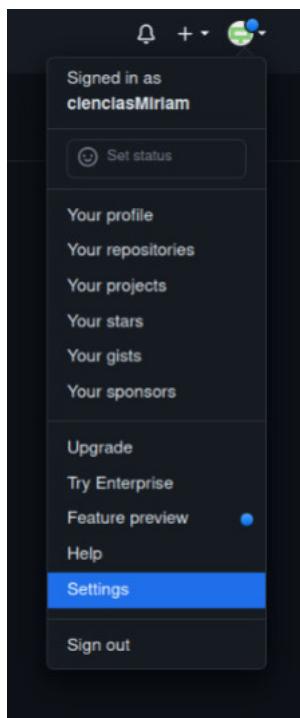


Figura 1: Perfil GitHub.

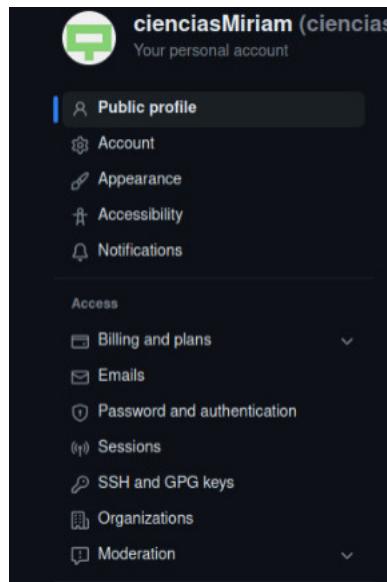


Figura 2: Opciones.

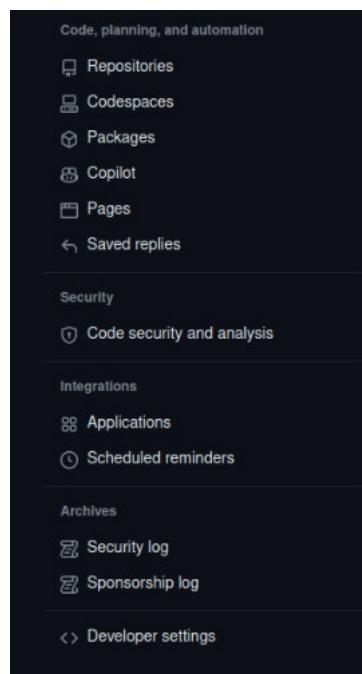


Figura 3: Opciones.

**Nota:** El token que vamos a generar es un [token de acceso personal detallados](#).



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Nos saldrá lo siguiente en pantalla:

The screenshot shows the GitHub developer settings page under 'Personal access tokens'. The 'Fine-grained tokens' option is selected. A red box highlights the 'Personal access tokens' dropdown menu. A blue box highlights the 'Generate new token' button at the top right.

Aquí, lo primero que debemos hacer es elegir el token que vamos a generar, esto se encuentra marcado en el recuadro color rojo. Daremos la primer opción que nos dice [Fine-grained tokens](#).

Después, daremos click en la opción que se encuentra marcada en el recuadro azul, que dice [Generate new token](#) y nos pedirá nuestra contraseña para autenticarnos.

- Lo siguiente será configurar nuestro token.

The screenshot shows the 'New fine-grained personal access token' configuration page. It includes fields for 'Token name' (MyFirstToken), 'Expiration' (30 days), 'Description' (Mi primer token), and a 'Repository access' section with three options: 'Public Repositories (read-only)', 'All repositories', and 'Only select repositories'.

Le asignamos un nombre, en este caso, le pusimos de nombre [MyFirstToken](#), así como la descripción; dicha información están en el recuadro color rojo.

Por otro lado, tiene una expiración de 30 días, es decir, este token nos será útil por ese período de tiempo, pasados los 30 días tendremos que generar uno nuevo si es que lo necesitamos.



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Este token, lo podemos usar en los repositorios públicos, privados o ambos, esto depende del uso que le demos. Aquí, lo configuramos para los repositorios que son públicos.

Lo siguiente es configurar el acceso de permisos del token los cuales se muestran a continuación:

Permissions

Read our [permissions documentation](#) for information about specific permissions.

**Account permissions**

User permissions permit access to resources under your personal GitHub account.

**Block another user** ⓘ View and manage users blocked by the user.

Select an access level

No access  
Read-only  
Read and write

**Codespaces user secrets** ⓘ Manage Codespaces user secrets.

**Email addresses** ⓘ Manage a user's email addresses.

**Followers** ⓘ A user's followers.

**GPG keys** ⓘ View and manage a user's GPG keys.

**Gists** ⓘ Create and modify a user's gists and comments.

Figura 4: Permisos de acceso.

Git SSH keys ⓘ

Interaction limits ⓘ

Plan ⓘ

Private repository invitations ⓘ

Profile ⓘ

SSH signing keys ⓘ

Starring ⓘ

Watching ⓘ

Access: No access  
Access: No access

Figura 5: Permisos de acceso.

Es **importante** que configuremos dichos permisos de acceso y elijamos el nivel de acceso los cuales son:

- Ningún acceso.
- Acceso de solo lectura.
- Acceso de lectura y escritura.

Puedes configurar, por el momento, que todos los permisos sean de acceso de lectura y escritura, sin embargo, es recomendable que [leas la documentación de GitHub](#) para que tengas más información al respecto.

Por último, nos da la información de los 14 permisos que le asignamos al token y finalmente damos click en el botón verde que dice [Generate token](#).

Overview

0 permissions for none of your repositories

14 Account permissions

This token will expire **July 21, 2023**.

Generate token Cancel

This token will be ready for use immediately.



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Finalmente, nuestro token ha sido generado y se encuentra en el recuado color morado.

The screenshot shows the GitHub developer settings page under 'Personal access tokens'. It displays a single token entry for 'Fine-grained tokens' with the following details:

- Name: Never used
- Token Value: `github_pat_11BATJR2Y0L13mC1Vbdex6_jMborl9TYZ3Rr1wb3` (highlighted with a purple box)
- Expiration: Expires on Thu, Jul 20 2023
- Actions: Copy icon (highlighted with a purple box), Delete button

**Nota:** Es importante que guardes el token en un lugar seguro y que este a la mano pues éste solo se muestra una vez y en caso de perderlo, tendrías que generar otro.



## 5.5. ¿Cómo clonar un repositorio y subir cambios en GitHub?

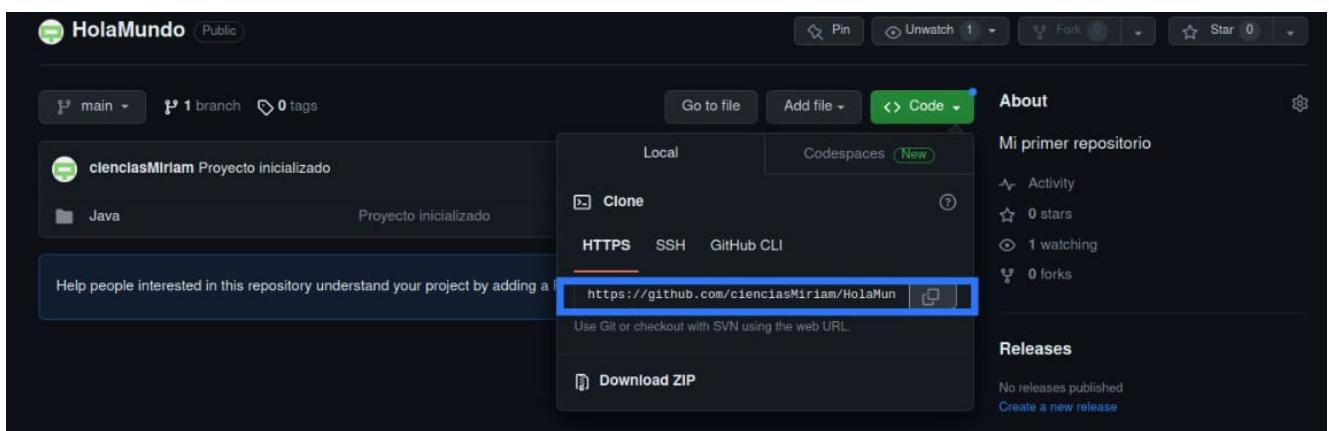
Ya vimos como crear un repositorio desde cero y subir un proyecto a él. Ahora, veremos como descargar un proyecto, desde la terminal, que ya está en la plataforma de GitHub para empezar a trabajar con los cambios que este ya tiene y añadirle mas funcionalidades.

Seguimos los siguientes pasos:

- Abrimos la terminal y nos posicionamos en la carpeta donde queremos iniciar la descarga del repositorio. En nuestro caso, será en la carpeta Descargas.

```
$ cd Descargas
```

- Ahora, iremos al repositorio que queremos descargar. Ahí encontraremos un botón verde que dice <>Code daremos click en él y vamos a copiar la liga HTTPS, marcada en el recuadro color azul, para poder clonar o descargar el repositorio.



- Ya teniendo la liga para clonar el repositorio, vamos a ejecutar el comando `git clone` seguido de la liga del repositorio.

```
$ git clone https://github.com/cienciasMiriam/HolaMundo.git
```

La salida en terminal es similar a la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas$ git clone https://github.com/cienciasMiriam/HolaMundo.git
Clonando en 'HolaMundo'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0
Recibiendo objetos: 100% (5/5), listo.
fciencias@fciencias-VirtualBox:~/Descargas$
```



# Facultad de Ciencias

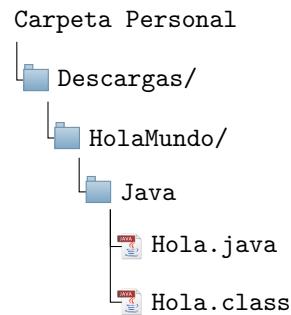
---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Entonces, en nuestra carpeta Descargas tendrá que existir la carpeta [HolaMundo](#) que es nuestro repositorio junto con los archivos del proyecto que existen.

Algo así tendremos en nuestra carpeta:

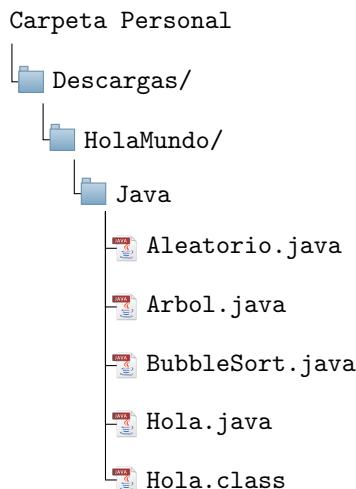


Listo, ya tenemos descargado el repositorio con en el que debemos trabajar.

Ahora, veremos como subir cambios al repositorio.

En la sección [¿Cómo subir un proyecto a GitHub?](#) se mencionan algunos comandos que tuvimos que ejecutar para poder subir los cambios, sin embargo, podemos omitir algunas palabras en los comandos para que sea más rápido el proceso.

Supongamos que ya trabajamos en los cambios del proyecto que descargamos y añadimos tres archivos [.java](#), nuestra carpeta ahora se ve así:



Para subir los cambios a nuestra plataforma de GitHub, haremos los siguientes pasos:

- Abrimos nuestra terminal y nos posicionamos en la carpeta del repositorio. Es decir, en nuestro caso, sería en la carpeta [HolaMundo](#).

```
$ cd Descargas/HolaMundo
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

- Inicializamos el repositorio ejecutando el siguiente comando:

```
$ git init
```

La salida en consola debe ser similar a la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git init
Reinicializado el repositorio Git existente en /home/fciencias/Descarga
s/HolaMundo/.git/
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ █
```

- Añadimos todos los archivos que tenemos dentro de la carpeta. En este paso, no importa que volvamos a añadir el archivo [Hola.java](#) y [Hola.class](#), no pasa nada. Por lo tanto lo podemos ejecutar sin problemas.

```
$ git add .
```

Este comando no produce ninguna salida en consola.

- Lo siguiente será hacer un [commit](#) y especificar los cambios que se efectuaron. Esto, recordemos que ayuda al trabajo en equipo pues es más visible de ver los cambios efectuados.

```
$ git commit -m "Se subieron 3 archivos nuevos"
```

Aquí nos muestra los cambios que se hicieron:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git commit -m "Se
subieron 3 archivos nuevos"
[main 467ef36] Se subieron 3 archivos nuevos
 3 files changed, 106 insertions(+)
  create mode 100644 Java/Aleatorio.java
  create mode 100644 Java/Arbol.java
  create mode 100644 Java/BubbleSort.java
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

- Finalmente subimos los cambios al repositorio. Recuerda que como contraseña debes poner el token que generaste para el repositorio.

```
$ git push
```

La salida en consola es la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git push
Username for 'https://github.com': cienciasMiriam
Password for 'https://cienciasMiriam@github.com':
Enumerando objetos: 8, listo.
Contando objetos: 100% (8/8), listo.
Compresión delta usando hasta 3 hilos
Comprimiendo objetos: 100% (5/5), listo.
Escribiendo objetos: 100% (6/6), 1.48 KiB | 1.48 MiB/s, listo.
Total 6 (delta 0), reusados 0 (delta 0), pack-reusados 0
To https://github.com/cienciasMiriam/HolaMundo.git
    a84ea94..467ef36 main -> main
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ █
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

- Vamos al repositorio y observemos que el commit que hicimos está hecho y está enmarcado en el recuadro color rojo.

The screenshot shows a GitHub repository named 'HolaMundo'. The commit message is 'Se subieron 3 archivos nuevos'. Below the message, there is a list of files: 'Java' (highlighted with a blue box) and 'Se subieron 3 archivos nuevos' (highlighted with a red box). The commit was made by 'cienciasMiriam' 7 minutes ago, with 2 commits in total. The repository has 0 stars and 1 watching. There is a button to 'Add a README'.

- Podemos dar click en el commit y podremos ver los archivos que subimos al repositorio junto con su contenido.

The screenshot shows the commit details for 'Se subieron 3 archivos nuevos'. The commit was made by 'cienciasMiriam' 8 minutes ago. It shows 3 changed files with 106 additions and 0 deletions. The files listed are 'Java/Aleatorio.java', 'Arbol.java', and 'BubbleSort.java'. The code editor shows the content of 'Java/Aleatorio.java'.

```
22 Java/Aleatorio.java
1 + /**
2 + * Importamos el paquete correspondiente.
3 + */
4 + import java.util.Random;
5 +
6 + * Clase Aleatorio.
7 + * Nos imprime un arreglo de 7 elementos con valores aleatorios.
8 + * @author Miriam Torres Bucio.
9 +
10 +public class Aleatorio{
11 +    /**
12 +     * Método main.
13 +     */
14 +    public static void main(String[] args){
15 +        int[] aleatorio = new int[7];
16 +        for(int i=0; i<7; i++){
17 +            aleatorio[i] = (int)(Math.random()*100);
18 +            System.out.println("[" + (i+1) + "] = " + aleatorio[i]);
19 +        }
20 +        System.out.println();
21 +    }
22 +}
```

Observemos que en esta parte solo aparecen los archivos nuevos que subimos, los cuales están enmarcados en el recuadro color rojo.



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Por último, si queremos verificar que no se borraron los demás archivos que teníamos en el repositorio, podemos ir a la carpeta que dice **Java** que está enmarcada en un recuadro color azul, al inicio de nuestro repositorio. Ahí, encontraremos todos los archivos.

The screenshot shows a GitHub repository named 'HolaMundo'. In the 'Code' tab, there is a 'Java' folder highlighted with a red box. Inside the 'Java' folder, there are four files: 'Aleatorio.java', 'Arbol.java', 'BubbleSort.java', and 'Hola.class'. Below the 'Java' folder, there is a table of commits. The first three commits, which correspond to the files in the 'Java' folder, are highlighted with a blue box. Each commit has a message like 'Se subieron 3 archivos nuevos' and a date like '9 minutes ago'. The last two commits, 'Hola.class' and 'Hola.java', are not highlighted and have messages like 'Proyecto inicializado' and dates like 'yesterday'.

Name	Last commit message	Last commit date
..		
Aleatorio.java	Se subieron 3 archivos nuevos	9 minutes ago
Arbol.java	Se subieron 3 archivos nuevos	9 minutes ago
BubbleSort.java	Se subieron 3 archivos nuevos	9 minutes ago
Hola.class	Proyecto inicializado	yesterday
Hola.java	Proyecto inicializado	yesterday

Observemos que todos los archivos se encuentran disponibles y están enmarcados en el recuadro color rojo.

Por otro lado, nos da la información de cada archivo, por ejemplo, los nuevos que subimos los enmarca con su respectivo commit, así como la fecha en la que se subieron dichos cambios.

Listo. Con esto podemos seguir trabajando en nuestro proyecto.

**Nota:** Es importante que al trabajar en equipo se sincronicen para poder hacer bien los cambios pues pueden existir errores al momento de hacer subirlos pues puede que alguien no haya descargado los cambios anteriores, subir nuevos y esto de forma automática estaría borrando lo que en un principio no descargó.



### 5.6. ¿Cómo crear ramas y cambiarlas en GitHub?

Las ramas de trabajo es donde se encuentra alojado nuestro proyecto en el repositorio que tengamos asignado. Podemos tener más de una rama en un repositorio.

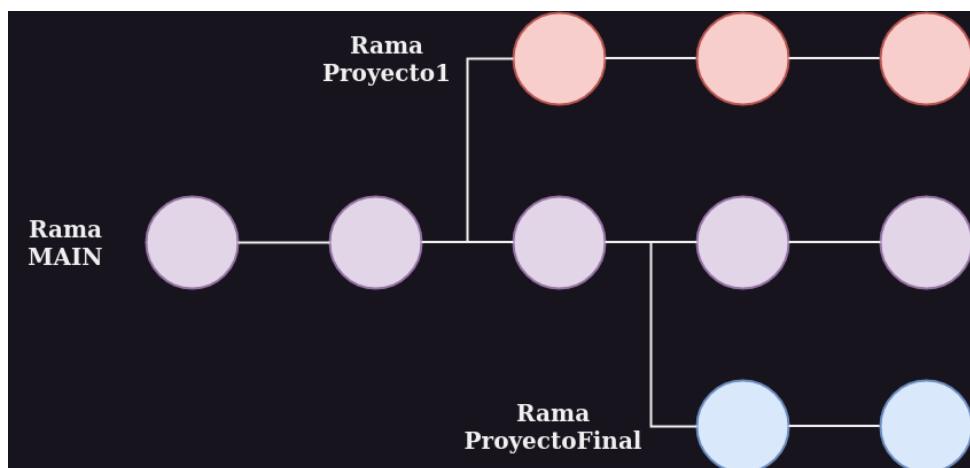
Las ramas son de gran utilidad al momento de subir un proyecto en un repositorio que tengamos en GitHub. Su uso principal es alojar en distintas ramas partes de nuestro proyecto en equipo y tener una mejor distribución de el y al final juntarlo todo.

Por otro lado, por ejemplo, si ocurre un error al momento de hacer un [push](#) trabajando en equipo, podemos crear otra rama de trabajo y solucionar el error.

Supongamos que estamos en la rama [main](#), cada punto nos dice que hubo un cambio en esa rama.

Pues podemos crear otra rama que se llame [Proyecto1](#) y trabajar en ella en alguna modificación del código por ejemplo.

Se pueden crear distintas ramas y cada una tendrá código diferente, por ejemplo, en la imagen existen tres ramas.



Para crear ramas y movernos a ellas para empezar a trabajar, hacemos los siguientes pasos:

- El primer paso que debemos efectuar al estar trabajando con git, siempre será [iniciar](#) el repositorio. Entonces, ejecutamos el comando:

```
$ git init
```

Recuerda que debes estar posicionado en la carpeta del repositorio.

- Para checar en qué rama estamos trabajando, vamos a ejecutar el siguiente comando:

```
$ git branch
```

- Por otro lado, si queremos ver todas las ramas con las que hemos trabajado y su seguimiento, ejecutamos el siguiente comando:

```
$ git branch -a
```



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Por ejemplo, en esta imagen nos dice que solo hemos trabajado con la rama `main`.

**Nota:** Otra forma de saber en qué rama estamos 'parados' es gracias al carácter `{*}` pues su función es aparecer del lado izquierdo de la rama en la que estamos trabajando.

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git branch
* main
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

- Ahora, vamos a crear una nueva rama de trabajo. Para esto solo basta con ejecutar el comando `branch` seguido del nombre de la rama.

```
$ git branch Proyecto1
```

Después, solo para verificar que se hizo la rama, ejecutamos nuevamente:

```
$ git branch -a
```

La salida en consola es la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git branch Proyec
to1
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git branch -a
  Proyecto1
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

- Ya creada la rama, es momento de `movernos` a ella para empezar a trabajar y subir los cambios que queramos, para esto, ejecutamos el comando:

```
$ git checkout Proyecto1
```

La salida en consola es la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git checkout Proj
ecto1
Cambiado a rama 'Proyecto1'
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Si queremos, podemos verificar que efectivamente estamos parados sobre la rama [Proyecto1](#) ejecutando el comando:

```
$ git branch
```

La salida en consola es la siguiente:

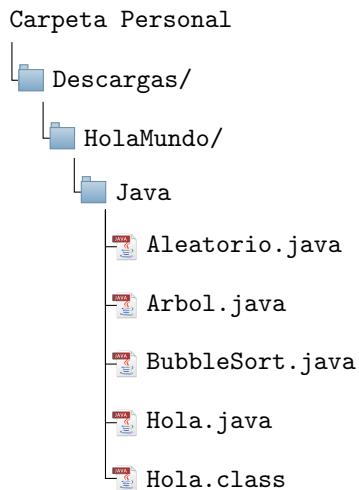
```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git branch
* Proyecto1
  main
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

Y listo, ya hemos creado una nueva rama y podemos trabajar con ella.

**Nota:** Si quieras ahorrar un paso al crear una rama y moverte a ella al mismo tiempo, puedes hacerlo ejecutando el comando:

```
$ git checkout -b NOMBRE-RAMA
```

- Por último, vamos a suponer que descargamos el repositorio con el que hemos estado trabajando, es decir:



Todos esos archivos solo estarán en la rama [main](#).



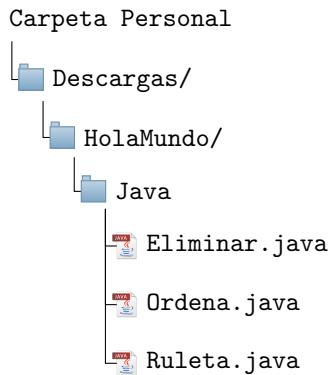
# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

Ahora, si vamos a trabajar en la rama [Proyecto1](#), subiremos el siguiente cambio. Aquí podemos eliminar los archivos que teníamos pues quedan guardados en la rama [main](#).



- Procedemos a subir los cambios, asegúrate de estar en la rama que creamos.

```
$ git add .  
$ git commit -m "Primer cambio en la rama Proyecto1"
```

Observemos que en este caso, nos dice qué elementos eliminamos y qué elementos insertamos.

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git add .  
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git commit -m "Primer cambio en la rama Proyecto1"  
[Proyecto1 0a4cbbd] Primer cambio en la rama Proyecto1  
 8 files changed, 127 insertions(+), 111 deletions(-)  
 delete mode 100644 Java/Aleatorio.java  
 delete mode 100644 Java/Arbol.java  
 delete mode 100644 Java/BubbleSort.java  
 create mode 100644 Java/Eliminar.java  
 delete mode 100644 Java/Hola.class  
 delete mode 100644 Java/Hola.java  
 create mode 100644 Java/Ruleta.java  
 create mode 100644 Java/ordena.java  
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

- Hacemos el [push](#) especificando en qué rama estamos haciendo el cambio.

```
$ git push --set-upstream origin Proyecto1
```



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

La salida en consola es la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git push --set-up
stream origin Proyecto1
Username for 'https://github.com': cienciasMiriam
Password for 'https://cienciasMiriam@github.com':
Enumerando objetos: 8, lista.
Contando objetos: 100% (8/8), lista.
Compresión delta usando hasta 3 hilos
Comprimiendo objetos: 100% (5/5), lista.
Escribiendo objetos: 100% (6/6), 1.57 KiB | 805.00 KiB/s, lista.
Total 6 (delta 0), reusados 0 (delta 0), pack-reusados 0
remote:
remote: Create a pull request for 'Proyecto1' on GitHub by visiting:
remote:     https://github.com/cienciasMiriam/HolaMundo/pull/new/Proye
cto1
remote:
To https://github.com/cienciasMiriam/HolaMundo.git
 * [new branch]      Proyecto1 -> Proyecto1
branch 'Proyecto1' set up to track 'origin/Proyecto1'.
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

Si vamos a nuestro repositorio, estará el siguiente cambio:

The screenshot shows a GitHub repository named 'HolaMundo'. At the top, there is a message: "Proyecto1 had recent pushes 18 minutes ago". Below this, there are buttons for "Pin", "Unwatch", "Fork", and "Star". The repository has 2 branches and 0 tags. The main branch is not protected. There is a message: "Your main branch isn't protected" with a link to "Protect this branch". A commit from user 'cienciasMiriam' is shown: "Se subieron 3 archivos nuevos" (2 days ago). The repository has 0 stars, 1 watching, and 0 forks. The "About" section says "Mi primer repositorio". The "Releases" section says "No releases published" and "Create a new release". The "Packages" section says "No packages published" and "Publish your first package".

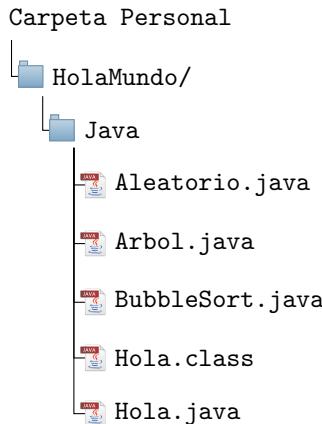
Y listo. Ahora existen dos ramas en el repositorio y cada una contiene archivos diferentes.



### 5.7. ¿Cómo descargar cambios en GitHub?

Al estar trabajando en equipo es normal que hayan cambios en el código que se escriba y si se está trabajando en una sola rama, hay que estar seguros de estar con la versión actualizada antes de hacer un [push](#) en el repositorio, es decir, subir cambios.

Por ejemplo, supongamos que tenemos la siguiente carpeta con archivos en el repositorio:



Estamos en la rama [\\*main](#) y podemos ver los archivos que tenemos en el repositorio si accedemos a la carpeta.

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ cd Java/
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo/Java$ ls
Aleatorio.java Arbol.java BubbleSort.java Hola.class Hola.java
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo/Java$ cd ..
```

Ahora, supongamos que hubo un cambio y es que se eliminó el archivo [Hola.class](#) del repositorio y además hubo cambio en el código.

**Nota:** Si se trabaja en equipo, es importante que antes de trabajar se descarguen los cambios que se hicieron para que no existan errores después o se lleguen a perder los avances.

Entonces, lo único que debemos hacer es ejecutar el siguiente comando:

```
$ git pull
```



# Facultad de Ciencias

---

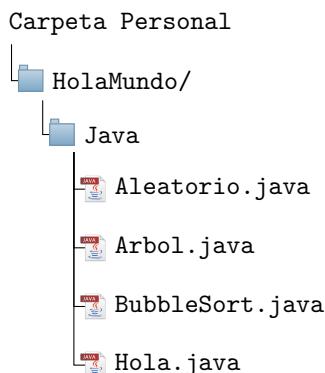
## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
Autor: Miriam Torres Bucio

La salida en la consola es la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git pull
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 11 (delta 4), reused 8 (delta 1), pack-reused 0
Desempaquetando objetos: 100% (11/11), 879 bytes | 125.00 KiB/s, listo.
Desde https://github.com/cienciasMiriam/HolaMundo
  90fd06c..761861a  main      -> origin/main
  0a4cbbd..72542ce  Proyecto1 -> origin/Proyecto1
Actualizando 90fd06c..761861a
Fast-forward
 Java/Hola.class | Bin 424 -> 0 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 Java/Hola.class
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

Los archivos quedan de la siguiente manera:



De esta manera, se han descargado los cambios efectuados y si revisamos la carpeta del repositorio el archivo ya no existe.

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ cd Java/
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo/Java$ ls
Aleatorio.java  Arbol.java  BubbleSort.java  Hola.java
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo/Java$
```



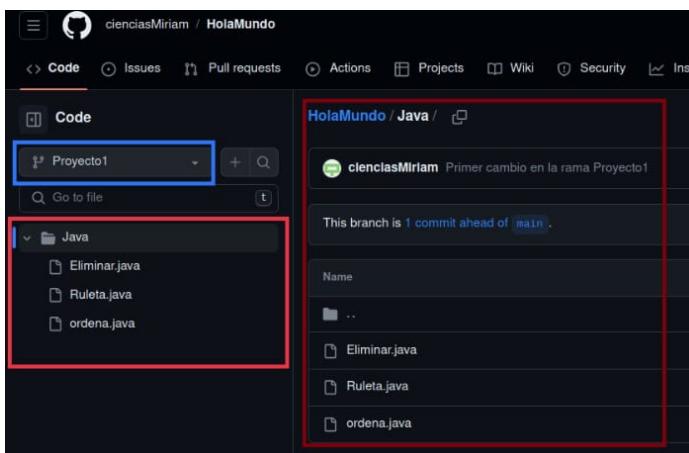
### 5.8. ¿Cómo mezclar varias ramas en un proyecto?

Hasta ahora, ya vimos como crear una nueva rama de trabajo y subir cambios en ella para poder trabajar sin interferir en otras ramas. De esta forma, si se trabaja en equipo, cada integrante podrá hacer la parte del proyecto que le corresponda sin interferir en el trabajo de los demás pues estará trabajando en su propia rama de trabajo.

Ahora, si cada persona hizo su parte del trabajo y todo funciona a la perfección, ¿cómo juntan su trabajo?

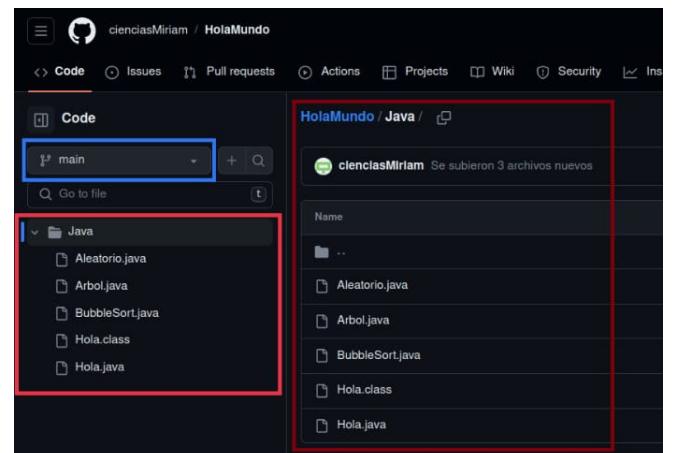
Git nos da la opción de mezclar el trabajo y poder hacer un [merge](#) en la rama de nuestra preferencia. Es importante que este proceso lo haga una sola persona para que no existan errores al momento de mezclar el código.

Las siguientes imágenes muestran el contenido individual de cada rama.



The screenshot shows the GitHub interface for the 'cienciasMiriam / HolaMundo' repository. The 'Code' tab is selected. A dropdown menu shows 'Proyecto1' is currently selected. Below it, a red box highlights the 'Java' folder. Inside the 'Java' folder, there are three files: 'Eliminar.java', 'Ruleta.java', and 'ordena.java'. To the right, a red box highlights the commit message: 'cienciasMiriam Primer cambio en la rama Proyecto1' and the status: 'This branch is 1 commit ahead of main.'

Figura 6: Rama Proyecto1.



The screenshot shows the GitHub interface for the 'cienciasMiriam / HolaMundo' repository. The 'Code' tab is selected. A dropdown menu shows 'main' is currently selected. Below it, a red box highlights the 'Java' folder. Inside the 'Java' folder, there are six files: 'Aleatorio.java', 'Arbol.java', 'BubbleSort.java', 'Hola.class', and two instances of 'Hola.java'. To the right, a red box highlights the message: 'cienciasMiriam Se subieron 3 archivos nuevos'.

Figura 7: Rama Main.

Supongamos que el proyecto está terminado, y que se le agregaron más funcionalidades a cada archivo y está listo para poder ser evaluado.

Vamos a mezclar el código en la rama [main](#) y seguimos los siguientes pasos:

- Abrimos nuestra terminal posicionándonos en la carpeta donde tenemos nuestro repositorio.
- Inicializamos nuestro repositorio.

```
$ git init
```

- Verificamos en qué rama estamos parados.

```
$ git branch
```

Recordemos que el [merge](#) lo queremos hacer en la rama [main](#), entonces, en consola nos tiene que aparecer:

```
Proyecto1
* main
```

De lo contrario, cambiamos de rama de trabajo como ya lo vimos con anterioridad.



# Facultad de Ciencias

---

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Ya posicionados en la rama main, lo siguiente es mezclarla con la rama Proyecto1. Ejecutamos el siguiente comando:

```
$ git merge Proyecto1
```

La salida en consola es la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git merge Proyecto1
Actualizando 467ef36..0a4cbbd
Fast-forward
 Java/Aleatorio.java | 22  -----
 Java/Arbol.java     | 34  -----
 Java/BubbleSort.java| 50  -----
 Java/Eliminar.java | 75 ++++++-----+
 Java/Hola.class    | Bin 424 -> 0 bytes
 Java/Hola.java      | 5  ---
 Java/Ruleta.java   | 19 ++++++-
 Java/ordena.java   | 33 ++++++-----+
 8 files changed, 127 insertions(+), 111 deletions(-)
 delete mode 100644 Java/Aleatorio.java
 delete mode 100644 Java/Arbol.java
 delete mode 100644 Java/BubbleSort.java
 create mode 100644 Java/Eliminar.java
 delete mode 100644 Java/Hola.class
 delete mode 100644 Java/Hola.java
 create mode 100644 Java/Ruleta.java
 create mode 100644 Java/ordena.java
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

Al hacer un merge de ambas ramas, ahora ya tenemos todos los archivos juntos. Si verificamos la carpeta, ahora podremos ver los archivos fusionados.

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ cd Java/
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo/Java$ ls
Aleatorio.java  BubbleSort.java  Hola.java   Ruleta.java
Arbol.java      Eliminar.java   ordena.java
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo/Java$
```

- Por último, subimos los cambios a la rama `main` por si queremos tener los cambios guardados.

Ejecutamos los siguientes comandos:

```
$ git add .
$ git commit -m "Merge completado"
$ git push
```



# Facultad de Ciencias

## UNAM

Ciencias de la Computación  
Universidad Nacional Autónoma de México  
Sistema Linux  
**Autor:** Miriam Torres Bucio

- Ya hechos los cambios, podemos ir al repositorio y verificar que se hizo de forma correcta el push de los archivos en nuestra rama.

Your main branch isn't protected

Protect this branch

TorresMiriam Merge completo

Java

Merge completo

94c2d3a 12 minutes ago 8 commits

12 minutes ago

Add a README

About

Mi primer repositorio

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Observemos que nuestro commit se hizo de forma correcta y está marcado el el recuadro color azul.  
Por otro lado, si vamos a nuestra carpeta Java, el recuadro de color rojo, podremos ver los archivos que mezclamos de la rama Proyecto uno junto con la rama main.

cienciasMiriam / HolaMundo

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Code

main

Go to file

Java

Aleatorio.java

Arbol.java

BubbleSort.java

Eliminar.java

Hola.java

Ruleta.java

ordena.java

TorresMiriam Merge completo

Name

Aleatorio.java

Arbol.java

BubbleSort.java

Eliminar.java

Hola.java

Ruleta.java

ordena.java



## 5.9. ¿Cómo eliminar una rama en GitHub?

Podemos eliminar una rama en GitHub siempre que lo deseemos o lo veamos necesario. Por ejemplo, si ya hicimos un merge de dos ramas, podemos eliminar la que no tiene todos los archivos.

En el repositorio con el que hemos estado trabajando, hicimos el merge en la rama `*main` por lo que en esa rama se quedó el proyecto finalizado. Ahora bien, si lo deseamos, podemos eliminar la rama `Proyecto1`.

Lo primero que debemos verificar es que estemos posicionados en la rama que contiene todo el proyecto.

```
$ git branch
```

La salida en consola debe ser la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git branch
  Proyecto1
* main
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ █
```

En caso de estar parados en otra rama, simplemente cambiamos de posición.

Ahora, para eliminar la rama `Proyecto1`, ejecutamos el siguiente comando:

```
$ git branch -D Proyecto1
```

La salida en consola es la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git branch -D Pro
yecto1
Eliminada la rama Proyecto1 (era 90fd06c).
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$
```

Y si ejecutamos el comando para ver las ramas de nuestro repositorio ya no nos dirá la opción de la rama `Proyecto1`.

```
$ git branch
```

La salida es la siguiente:

```
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ git branch
* main
fciencias@fciencias-VirtualBox:~/Descargas/HolaMundo$ █
```

Listo, de esta forma podemos eliminar ramas que ya no necesitamos.