



Universidad de las Fuerzas Armadas - ESPE

Pregrado SII OCT24 – MAR25

Programación Orientada a Objetos – Nrc: 1323

Control de Lectura GitHub

Nombre del estudiante:

David Ivan Granada Pachacama

Nombre del docente:

Mgtr. Luis Enrique Jaramillo Montaña

Fecha de entrega: 20 de noviembre del 2024

Contenido

Introducción.....	4
1. Objetivos.....	4
1.1. Objetivos General.....	4
1.2. Objetivos Específicos	4
2. VCS	4
3. Git y GitHub	5
4. Cuadro comparativo Git y GitHub	9
5. Resultados.....	18
6. Link del repositorio en GitHub	19
7. Creación de un archivo local.....	19
8. Consulta No. 1	24
9. Consulta No. 2	27
10. Conclusiones	29
11. Recomendaciones	29
12. Referencias Bibliográficas.....	30

Tabla de figuras

Figura 1. VCS funcionamiento	5
Figura 2. Como eliminar archivos	6
Figura 3. Teclas Ctrl+z	6
Figura 4. Personas preocupadas	7
Figura 5. Prototipó de una máquina del tiempo.....	7
Figura 6. Repositorio GitHub.	8
Figura 7. GitHub funcionamiento interno.....	8
Figura 8. Git instalado en Windows 10.....	11
Figura 9. Protocolo de SSH en Windows 10	12
Figura 10. Ejemplo de clase Padre e Hija	26

Introducción

El sistema de control de versionamiento es una herramienta fundamental para el desarrollo del software moderno ya que nos permite gestionar y rastrear los cambios de los códigos fuentes a lo largo del tiempo. Estos sistemas en específico nos permiten facilitar la colaboración entre múltiples desarrolladores de software en la actualidad. En el presente informe exploraremos definiciones y técnicas para el correcto manejo de GitHub.

1. Objetivos

1.1. Objetivos General

- Comprender la relevancia del funcionamiento de los Sistemas de Control de Versionamiento (VCS) como herramienta local o vía online para la cooperación entre equipos, mejora la gestión de modificaciones en el código.

1.2. Objetivos Específicos

- Instalar el software Git en un sistema operativo.
- Crear un repositorio en GitHub vía online con la finalidad de recolectar información importante.
- Crear un archivo de manera local puede ser un código “Hola Mundo”, subir mediante línea de comandos a GitHub.

2. VCS

Los Sistemas de Control de Versionamiento (VCS) son herramientas y métodos utilizados para administrar y gestionar los diferentes cambios que se producen en el software o conjunto de ficheros, manteniendo un historial detallado de las modificaciones realizadas (Borrell Noguerras, 2006).

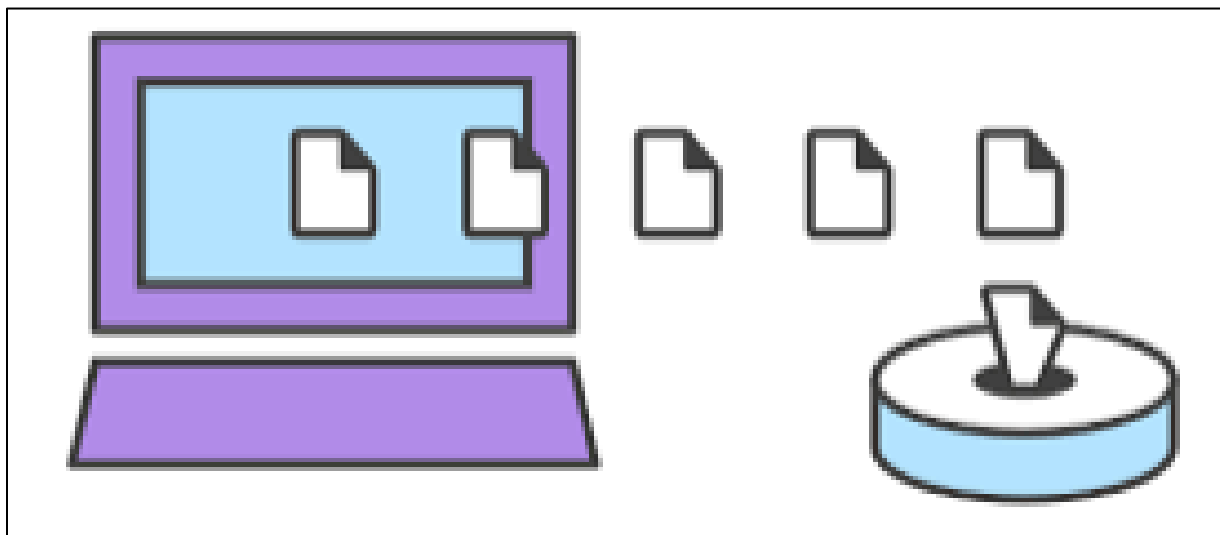
3. Git y GitHub

Un sistema de control de versiones es como una máquina mágica que ayuda a los equipos que hacen programas de computadoras a trabajar juntos y a no perder ningún cambio que hagan en sus programas. También les ayuda a volver atrás en el tiempo si algo sale mal y a resolver problemas si dos personas hacen cambios diferentes al mismo tiempo. Git y GitHub son como dos superhéroes que hacen esto muy bien. Git es como la herramienta que usamos para guardar y ver los cambios, y GitHub es como un lugar seguro en internet donde podemos guardar nuestros programas para que todos los demás puedan ayudar.

1. Un VCS, es un repositorio donde podemos guardar la información que utilizaremos en un proyecto como se puede observar en la figura 1.

Figura 1.

VCS funcionamiento



2. Qué pasa si realizamos una modificación o eliminación de un archivo sea de manera intencional o accidental, y queremos volver al archivo a como estaba antes, a continuación, se puede observar en la Figura 2.

Figura 2.

Como eliminar archivos



3. La mayoría diría ctrl+z, pero en caso que este archivo está guardado y no permita revertir el cambio a continuación, se puede observar en la Figura 3.

Figura 3.

Teclas Ctrl+z



4. Estaríamos realmente preocupados, en especial si se trata de un archivo muy importante, pues para eso nos sirve el VCS a continuación, se puede observar en la Figura 4.

Figura 4.

Personas preocupadas



5. Haz de referencia que con el VCS podrían utilizar una máquina del tiempo, y regresar al pasado y poder recuperar a continuación, se puede observar en la Figura 5.

Figura 5.

Prototipo de una máquina del tiempo.



6. En conclusión, se podría tener los archivos en un repositorio de manera segura a continuación, se puede observar en la Figura 6.

Figura 6.

Repositorio GitHub.



7. Además, podemos otorgar permisos a otras personas para que trabajen de manera simultánea, con la posibilidad de recuperar las versiones de los archivos cuando los queramos. Para el manejo del sistema de control de versión se utiliza una herramienta en Windows llamada Git, para el uso de este sistema de manera online, se utiliza plataforma web, la más conocida es GitHub a continuación, se puede observar en la Figura 7.

Figura 7.

GitHub funcionamiento interno.



4. Cuadro comparativo Git y GitHub

Tabla 1.

Cuadro comparativo Git y GitHub

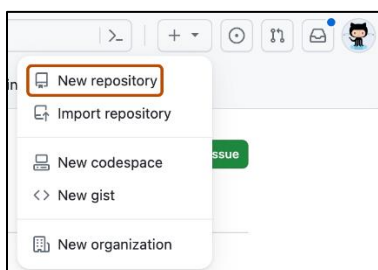
	Git	GitHub
Definición	Git es un sistema de control de versiones distribuido y de código abierto que se utiliza para rastrear los cambios en el código fuente de un proyecto de software.	GitHub es una plataforma en línea basada en la web que utiliza Git como sistema de control de versiones subyacente. Proporciona servicios de alojamiento de repositorios en línea y herramientas de colaboración para equipos de desarrollo.
Función principal:	Git se utiliza para gestionar y controlar el historial de versiones de un proyecto, lo que incluye realizar un seguimiento de los cambios, crear ramas de desarrollo, fusionar cambios y revertir a versiones anteriores.	GitHub se utiliza para alojar repositorios de Git de forma remota en servidores en línea. Permite a los equipos colaborar en proyectos, realizar seguimiento de problemas, revisar y aprobar cambios, y gestionar la colaboración de manera efectiva.
Ubicación:	Git se instala en tu computadora local y se utiliza para trabajar en un repositorio de código en tu máquina.	GitHub está en línea y aloja repositorios Git en servidores remotos. Los usuarios acceden a los repositorios y colaboran a través de la plataforma web de GitHub.
Acceso	Puedes usar Git sin estar conectado a internet. Es ideal para el desarrollo local y colaborativo en redes privadas.	Para trabajar con GitHub, necesitas una conexión a internet. Es ideal para colaboración en línea y proyectos de código abierto.

Creación de un repositorio en GitHub

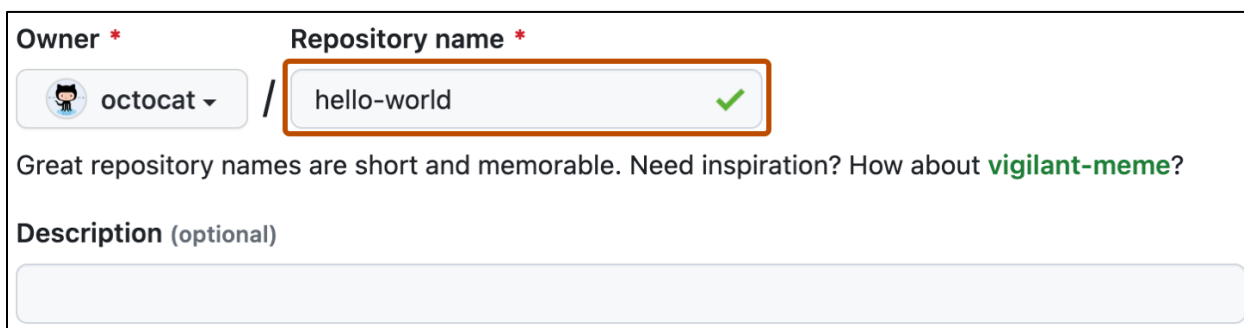
Según (*Inicio rápido para repositorios - Documentación de GitHub*, s. f., p. 1) concluye que:

Los repositorios GitHub pueden almacenar varios proyectos. En esta guía, creará un repositorio y confirmará el primer cambio.

1. En la esquina superior derecha de cualquier página, selecciona y luego haz clic en Nuevo repositorio.



2. Escriba un nombre corto y fácil de recordar para el repositorio. Por ejemplo: "hola-mundo".



3. Opcionalmente, puede agregar una descripción del repositorio. Por ejemplo, "Mi primer repositorio en GitHub".
4. Elige la visibilidad del repositorio. Para obtener más información, vea «Acerca de los repositorios».
5. Seleccione Initialize this repository with a README (Inicializar este repositorio con un archivo Léame).
6. Haga clic en Create repository (Crear repositorio).

Felicidades. Ha creado correctamente su primer repositorio y lo has inicializado con un archivo Léame.

Pasos para configurar la llave SSH en un sistema operativo.

1. Configurar Git en Windows

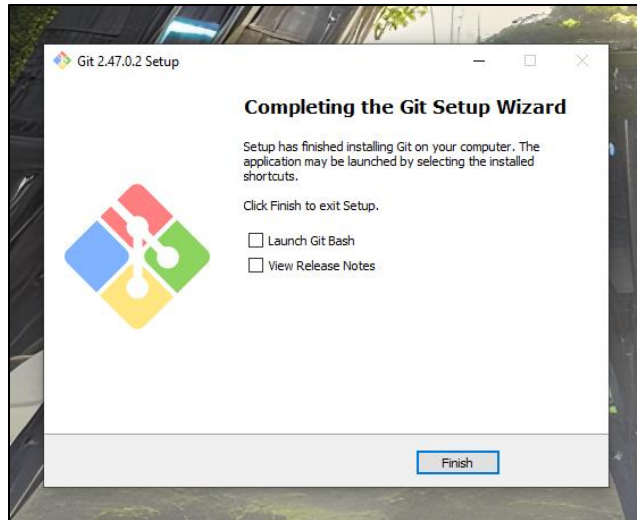
Para la instalación del Git en un sistema operativo Windows, se realiza el siguiente procedimiento.

- Ve al sitio web oficial de Git en <https://git-scm.com/download/win>.
- Descarga el instalador para Windows haciendo clic en el enlace "64-bit Git for Windows Setup" o "32-bit Git for Windows Setup" según tu versión de Windows.
- Ejecuta el archivo descargado para iniciar el instalador.

- En la primera pantalla del instalador, puedes dejar las opciones predeterminadas y hacer clic en "Next" (Siguiente)
 - Selecciona "Use Git from Git Bash only" para utilizar Git desde la línea de comandos de Git Bash o "Use Git from the Windows Command Prompt" si deseas utilizar Git desde la línea de comandos de Windows. Luego, haz clic en "Next" (Siguiente).
 - Selecciona "Use the OpenSSL library" y deja las demás opciones predeterminadas.
 - Haz clic en "Next" (Siguiente).
 - Selecciona "Checkout Windows-style, commit Unix-style line endings" y haz clic en "Next" (Siguiente).
 - Elige "Use Windows' default console window" y haz clic en "Next" (Siguiente).
 - En la pantalla "Select the default behavior of 'git pull'", elige "Rebase" y haz clic en "Next" (Siguiente).
 - Selecciona "Enable file system caching" y haz clic en "Next" (Siguiente).
 - Elige "Enable Git Credential Manager" y haz clic en "Next" (Siguiente).
 - En la pantalla "Configuring the line ending conversions", selecciona "Checkout as -is, commit Unix-style line endings" y haz clic en "Next" (Siguiente).
 - Haz clic en "Install" (Instalar) para comenzar la instalación de Git.
 - Espera a que se complete la instalación y luego haz clic en "Finish" (Finalizar).
- Instalación de GitHub.
- GitHub no se instala en tu computadora como una aplicación independiente. Es una plataforma en línea que se utiliza a través de un navegador web para alojar repositorios de código y colaborar en proyectos de desarrollo de software. A continuación, se observa en la Figura 8.

Figura 8.

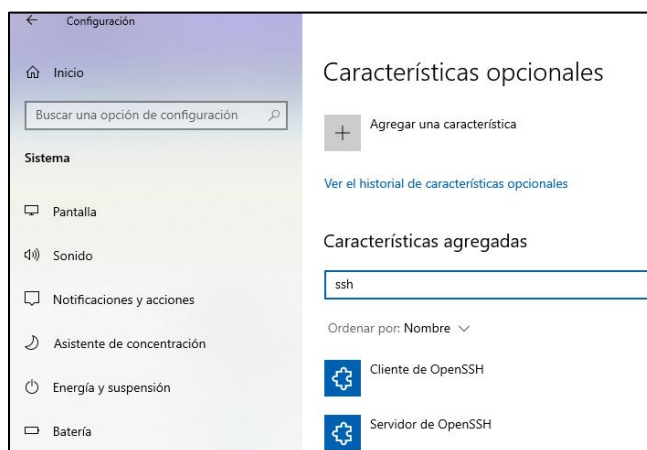
Git instalado en Windows 10



2. Instalar en Windows el protocolo SSH que son dos complementos como se puede observar en la Figura 9.

Figura 9.

Protocolo de SSH en Windows 10

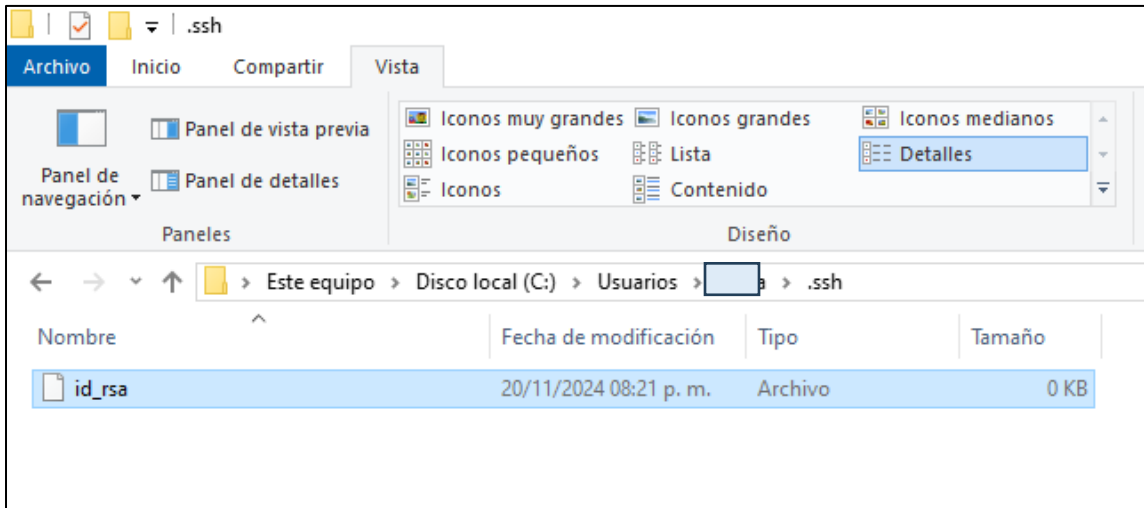


3. Crear una carpeta .ssh para guardar las llaves ssh para poder sincronizar en Git y GitHub.

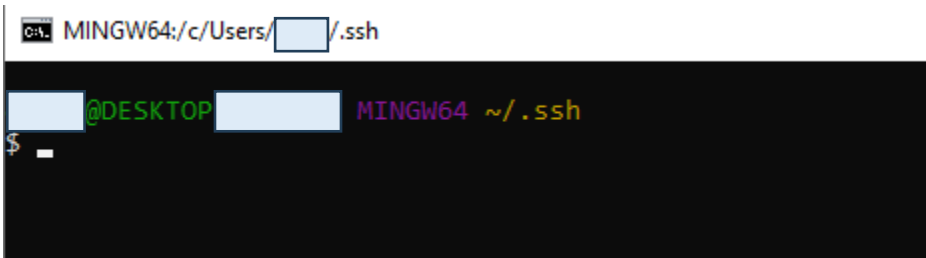
Para utilizar GitHub en el ordenador se debe realizar los siguientes pasos:

- a. Configurar el equipo para realizar la conexión con la plataforma de GitHub, mediante una conexión en este caso mediante el protocolo SSH.

- b. Crear una carpeta con el nombre de .ssh y un archivo con el nombre id_rsa para crear la llave pública.



- c. Abrir y ejecutar Git Bash previamente instalado en sistema operativo Windows.
- d. Ingresar por medio de consola a la carpeta donde esta el archivo creado para la llave ssh usando el siguiente comando. **cd C:/Users/nombre-usuario/.ssh**



- e. Escribir el siguiente código. **ssh-keygen.exe -o -t rsa -C**
“correo@dominio.com” el correo es con el que te registraste en GitHub, sobrecribir el archivo id_rsa ingresando **y** de yes. Finalmente crear una clave para la llave ssh y presionar Enter.

```

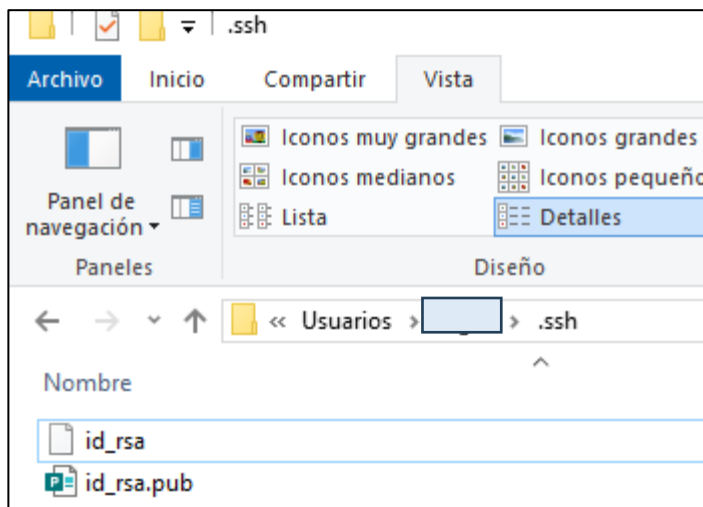
C:\ MINGW64:/c/Users/[redacted]/.ssh

@DESKTOP-[redacted] MINGW64 ~/.ssh
$ ssh-keygen.exe -o -t rsa -C "[redacted]@[redacted]"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/[redacted]/.ssh/id_rsa): id_rsa
id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase for "id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:AaNXnatdYo44hg1EF3S0iNcmgbk48TLbiICH/wuB67E "[redacted]@[redacted]"
The key's randomart image is:
+---[RSA 3072]---+
|  ..+Bo+.  . |
|  . ++ O .o |
| .. =O. = . |
| +.* +O O .+ . |
| o+.B + .S* o |
| ..+.O = O O |
| ..... . |
| . O.. |
| E .. |
+-----[SHA256]-----+

@DESKTOP-[redacted] MINGW64 ~/.ssh
$

```

f. Verificar que se haya creado dos archivos.



g. Leer la clave con el comando pública y copiarla. **cat id_rsa.pub**

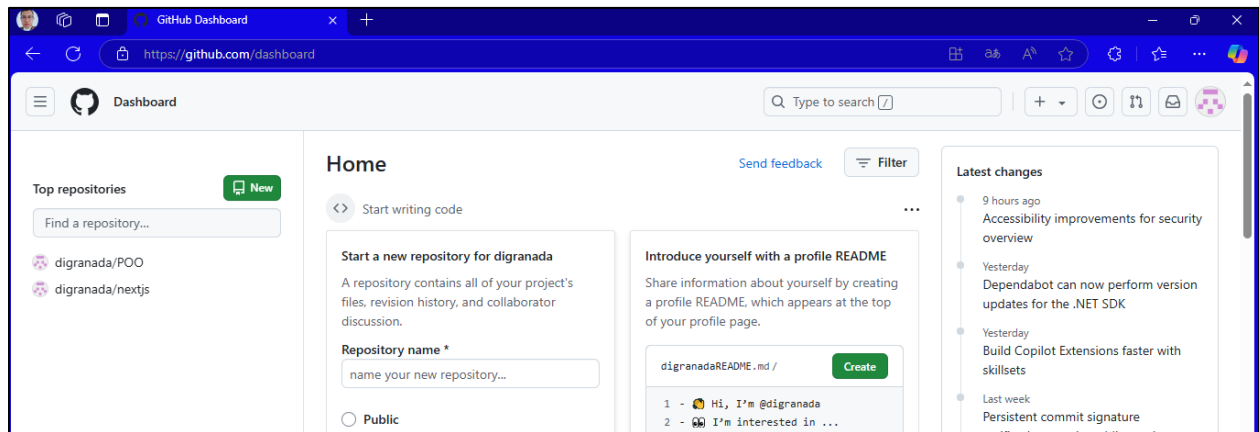
```

C:\ MINGW64:/c/U[redacted]/.ssh

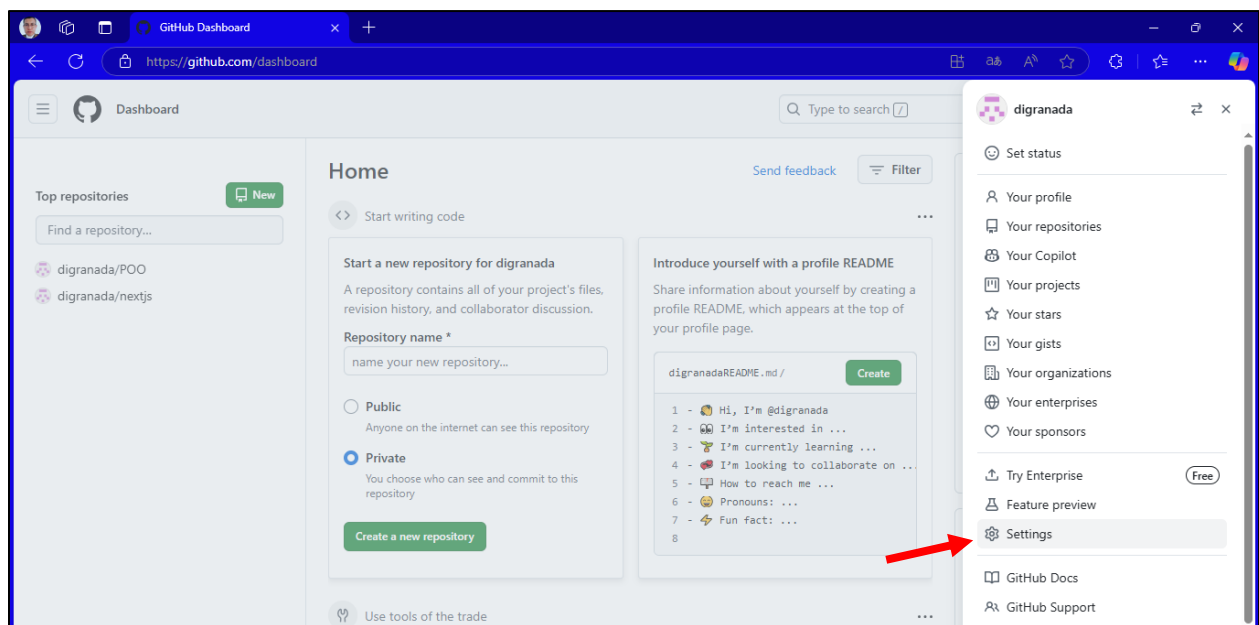
@DESKTOP-[redacted] MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAK0GCo1Q6FbkU7B9u0u7eDfma0CesVDu4K3cUaEVMVso9Cu8ZY6dd7f5vubukT0Pbd0c47HAM1610MwQBgZASbTge0CmIOX20kndUcjJqWd0VfTe1VpCz4/pCaxYX3XiqH1X9CBM2gKfy0ZDZ9C0D1fPS= d1granada@espe.edu.ec

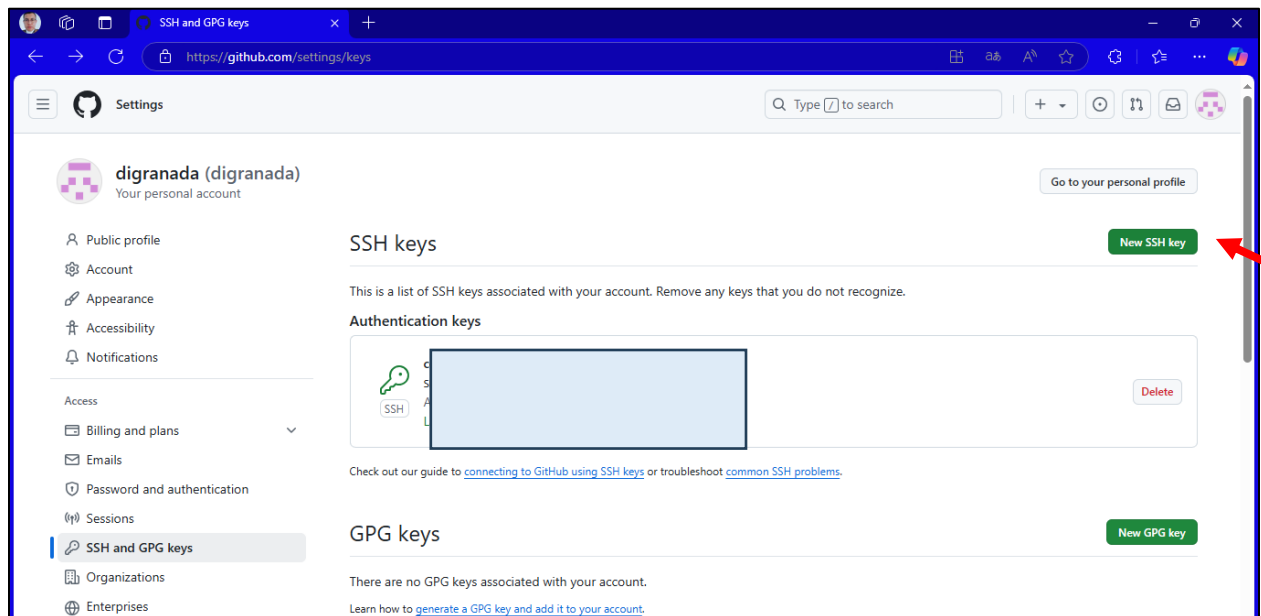
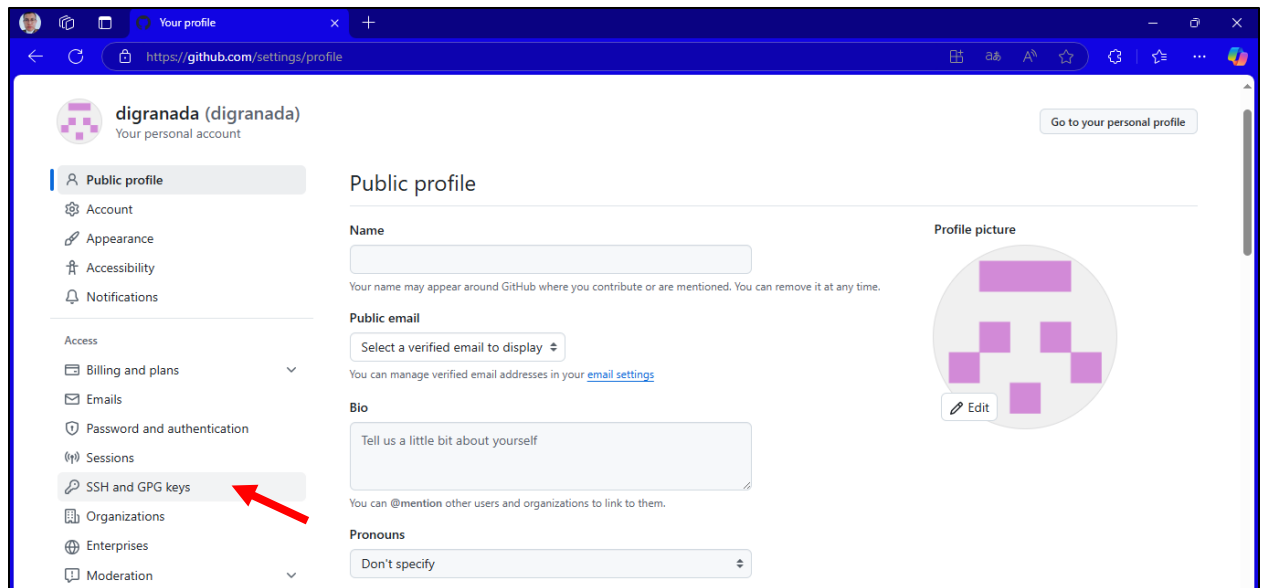
```

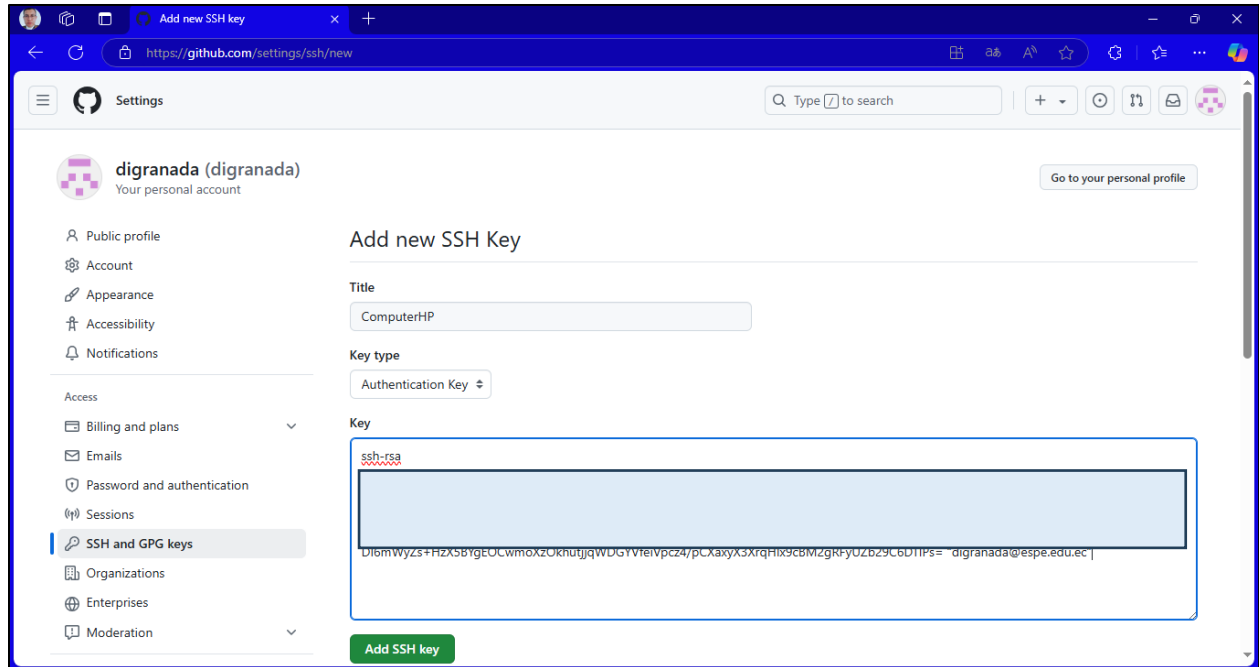
- h. Cargar llave a GitHub.
- Iniciar sesión de GitHub.



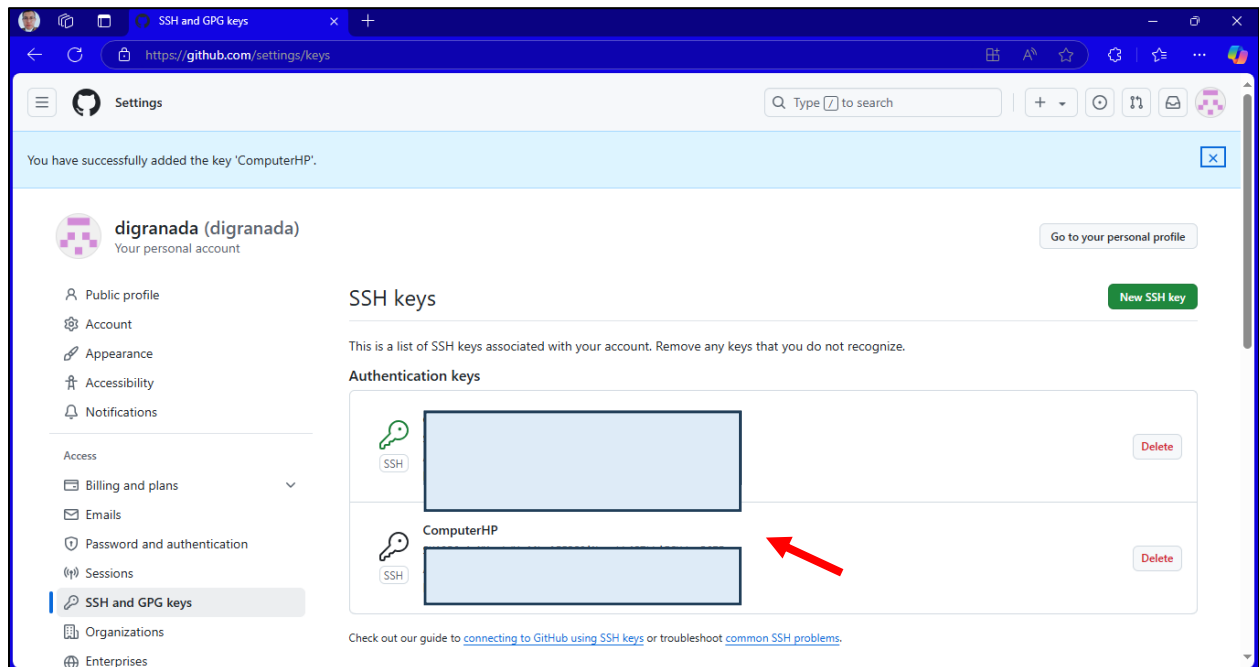
- Cargar la llave a GitHub, como se ve en las siguientes imágenes.







- Cargar las llaves conforme los siguientes pasos.



Finalizado estos pasos que permiten la conexión entre nuestro GitHub, y el equipo que utilizaremos podemos empezar a crear nuestro repositorio, si vas a manejar desde otro equipo adicional debes realizar nuevamente esta configuración, recuerda borrar las credenciales de equipos que no utilices.

5. Resultados


Creamos un nuevo repositorio dentro de GitHub

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 *

 digranada ▾

Repository name *


202451_1323

 202451_1323 is available.


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

Description (optional)

"La educación es el pasaporte hacia el futuro, el mañana pertenece a aquellos que se preparan para él hoy" - I

☒  **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.](#)

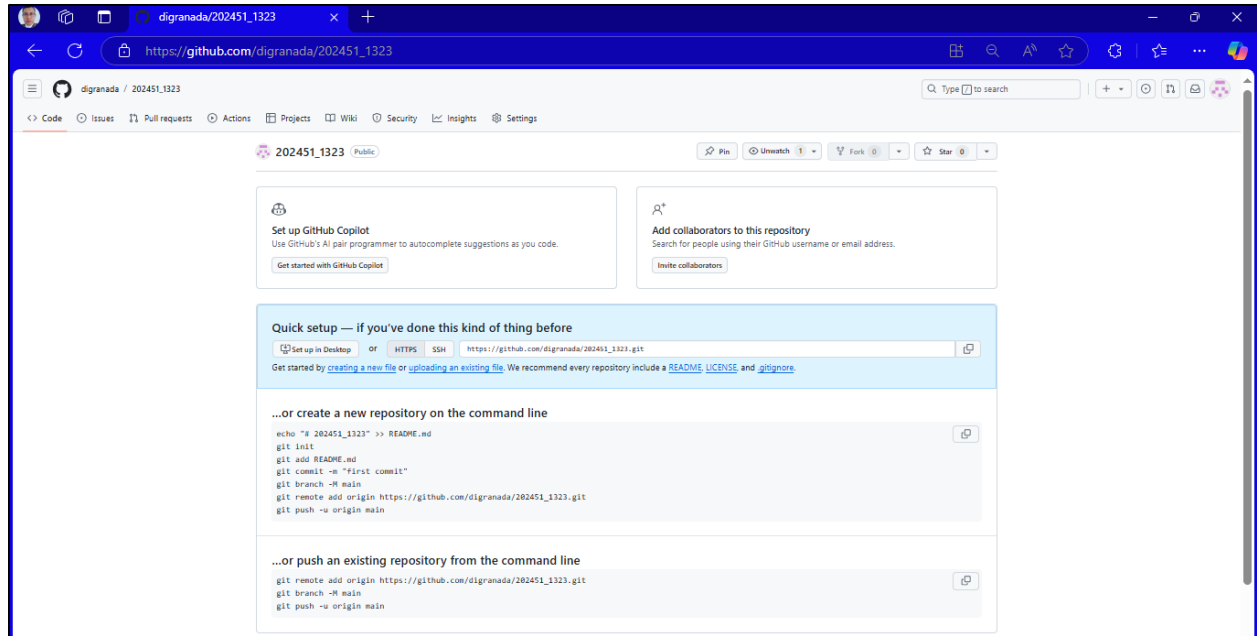
Choose a license

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



6. Link del repositorio en GitHub

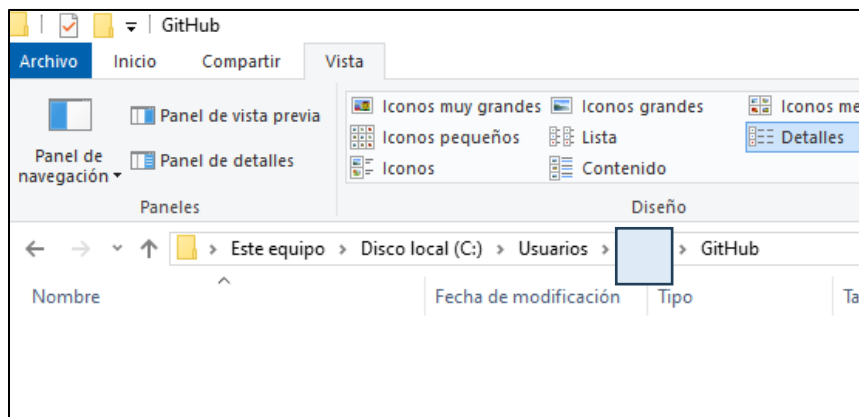
HTTPS: https://github.com/digranada/202451_1323.git

SSH: `git@github.com:digranada/202451_1323.git`

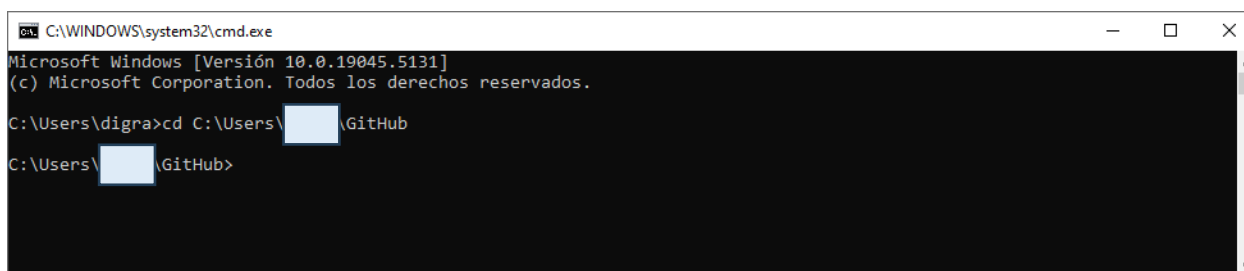
7. Creación de un archivo local

Una vez obtenida la información procedemos a la configuración en el ordenador con siguiendo los siguientes pasos.

Seleccionamos una carpeta que nos servirá para conectar con el repositorio



Accedemos a la carpeta desde la consola.



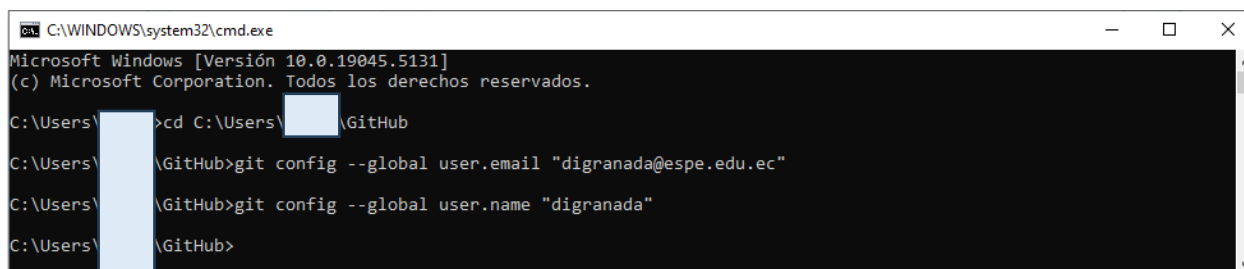
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\digra>cd C:\Users\ [redacted] \GitHub
C:\Users\ [redacted] \GitHub>
```

Ejecutamos los siguientes comandos.

```
git config --global user.email "@gmail.com"
```

```
git config --global user.name "Your user Name"
```



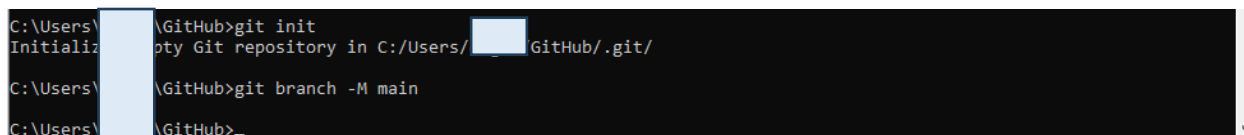
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\ [redacted] >cd C:\Users\ [redacted] \GitHub
C:\Users\ [redacted] \GitHub>git config --global user.email "digranada@espe.edu.ec"
C:\Users\ [redacted] \GitHub>git config --global user.name "digranada"
C:\Users\ [redacted] \GitHub>
```

Ejecutar los siguientes comandos.

```
git init
```

```
git branch -M main
```



```
C:\Users\ [redacted] \GitHub>git init
Initialized empty Git repository in C:/Users/[redacted]/GitHub/.git/

C:\Users\ [redacted] \GitHub>git branch -M main
C:\Users\ [redacted] \GitHub>
```

Los pasos obtenidos en la plataforma de github, copiamos y pegamos los comandos que faltarían.

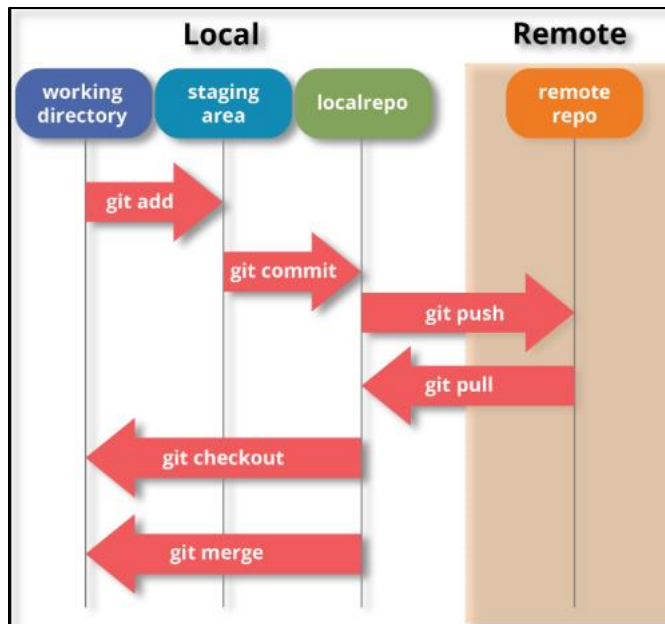
```
git remote add origin git@github.com:cushico/POO.git
```



```
C:\Users\ [redacted] \GitHub>git remote add origin git@github.com:digranada/202451_1323.git
C:\Users\ [redacted] \GitHub>
```

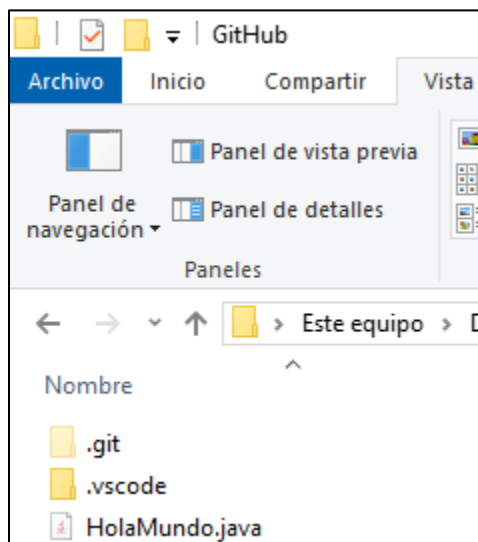
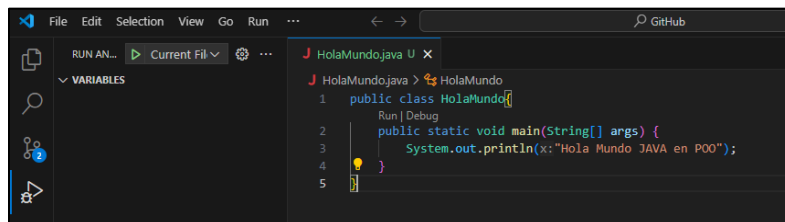
Una vez configurado, podemos realizar las pruebas para subir nuestro primer archivo a GitHub,

para lo cual tendremos que tener presente como se observa en la siguiente imagen.



Cargar un archivo a GitHub.

Creemos un archivo que utilizaremos para subir a GitHub.



Con la opción `git status -s` podemos observar el estado de los archivos.

?: 'sin seguimiento',

M = modificado

A = añadido

D = eliminado

R = renombrado

C = copiado

U = actualizado, pero no fusionado

```
PS C:\Users\ [redacted] \GitHub> git status -s
?? .vscode/
?? HolaMundo.java
PS C:\Users\ [redacted] \GitHub> |
```

Pasamos al staging area, como vimos en la imagen principal de GitHub, con el comando git add

```
PS C:\Users\digra\GitHub> git add HolaMundo.java
PS C:\Users\digra\GitHub> |
```

Para pasar al repositorio local, colocamos el comando.

```
PS C:\Users\ [redacted] \GitHub> git status -s
A HolaMundo.java
?? .vscode/
PS C:\Users\ [redacted] \GitHub> git commit -m "My first program upload to GitHub"
[main (root-commit) 3e789f7] My first program upload to GitHub
1 file changed, 5 insertions(+)
create mode 100644 HolaMundo.java
PS C:\Users\ [redacted] \GitHub> |
```

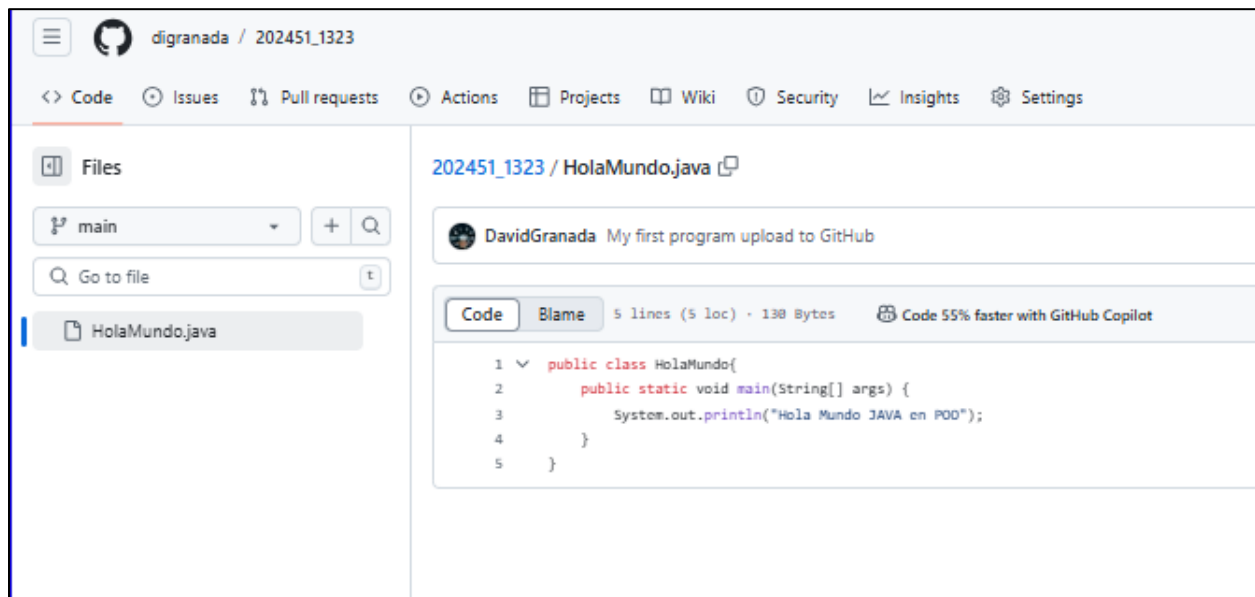
git commit "comentario que quiera colocar"

Para pasar del repositorio local a la nube, utilizamos el comando **git push -u origin**

Main

```
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 338 bytes | 112.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:digranada/202451_1323.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\ [redacted] \GitHub>
```

Una vez subida la información podemos comprobar en la plataforma.



8. Consulta No. 1

1. ¿Qué es Paradigma de Programación Orientado a Objetos?

Según La programación orientada a objetos (POO) es un paradigma de programación que se basa en el concepto de “objetos”, los cuales son instancias de clases. En POO, un “objeto” puede entenderse como una entidad que encapsula datos y funciones que operan sobre esos datos. Este enfoque permite modelar el mundo real de manera más efectiva, ya que se pueden representar entidades y sus interacciones de manera más natural.

2. ¿Qué es una clase?

En POO, una “clase” es un plano o plantilla para crear objetos. Define la estructura y el comportamiento de los objetos que se crearán a partir de ella.

3. ¿Qué es un atributo?

Los objetos son instancias concretas de una clase, y cada objeto puede tener sus propios datos (llamados atributos).

4. ¿Qué es un método?

Cada objeto puede tener sus propias funciones (llamadas métodos).

5. ¿Qué es un sistema de control de versionamiento y para qué sirve?

Según (*Git y GitHub - Aprende desarrollo web | MDN*, 2024, p. 1) concluye que:

Un sistema de control de versionamiento (VCS) una herramienta que les permita colaborar con otros desarrolladores en un proyecto sin peligro de que sobrescribir el trabajo de los demás, y volver a las versiones anteriores de la base de código si existe un problema descubierto más tarde. El VCS más popular (al menos entre los

desarrolladores web) es Git, junto con GitHub, un sitio que proporciona alojamiento para tus repositorios y varias herramientas para trabajar con ellos. Este módulo tiene como objetivo enseñarte lo que necesitas saber sobre ambos.

Los VCS son esenciales para el desarrollo de software:

Es raro que trabajes en un proyecto completamente por tu cuenta, y tan pronto como comiences a trabajar con otras personas, comenzarás a correr el riesgo de entrar en conflicto con el trabajo del otro, es decir, cuando ambos intentan actualizar simultáneamente la misma pieza de código. Debes tener algún tipo de mecanismo para administrar las ocurrencias y, como resultado, evitar la pérdida de trabajo.

Cuando trabajes en un proyecto por tu cuenta o con otros, querrás poder hacer una copia de seguridad del código en un lugar central, para que no se pierda si tu computadora se daña.

También querrás poder volver a versiones anteriores si más tarde descubres un problema. Es posible que hayas empezado a hacer esto en tu propio trabajo mediante la creación de diferentes versiones de un mismo archivo, por ejemplo `myCode.js`, `myCode_v2.js`, `myCode_v3.js`, `myCode_final.js`, `myCode_really_really_final.js`, etc, pero esto es muy propenso a errores y poco fiable.

Los diferentes miembros del equipo generalmente querrán crear sus propias versiones separadas del código (llamadas ramas en Git), trabajar en una nueva característica en esa versión y luego fusionarla de manera controlada (en GitHub usamos solicitudes de extracción) con la versión maestra cuando hayan terminado con ella.

Los VCS proporcionan herramientas para satisfacer las necesidades anteriores. Git es un ejemplo de VCS, y GitHub es un sitio web + infraestructura que proporciona un servidor Git más una serie de herramientas realmente útiles para trabajar con repositorios git individuales o en equipo, cómo informar problemas con el código,

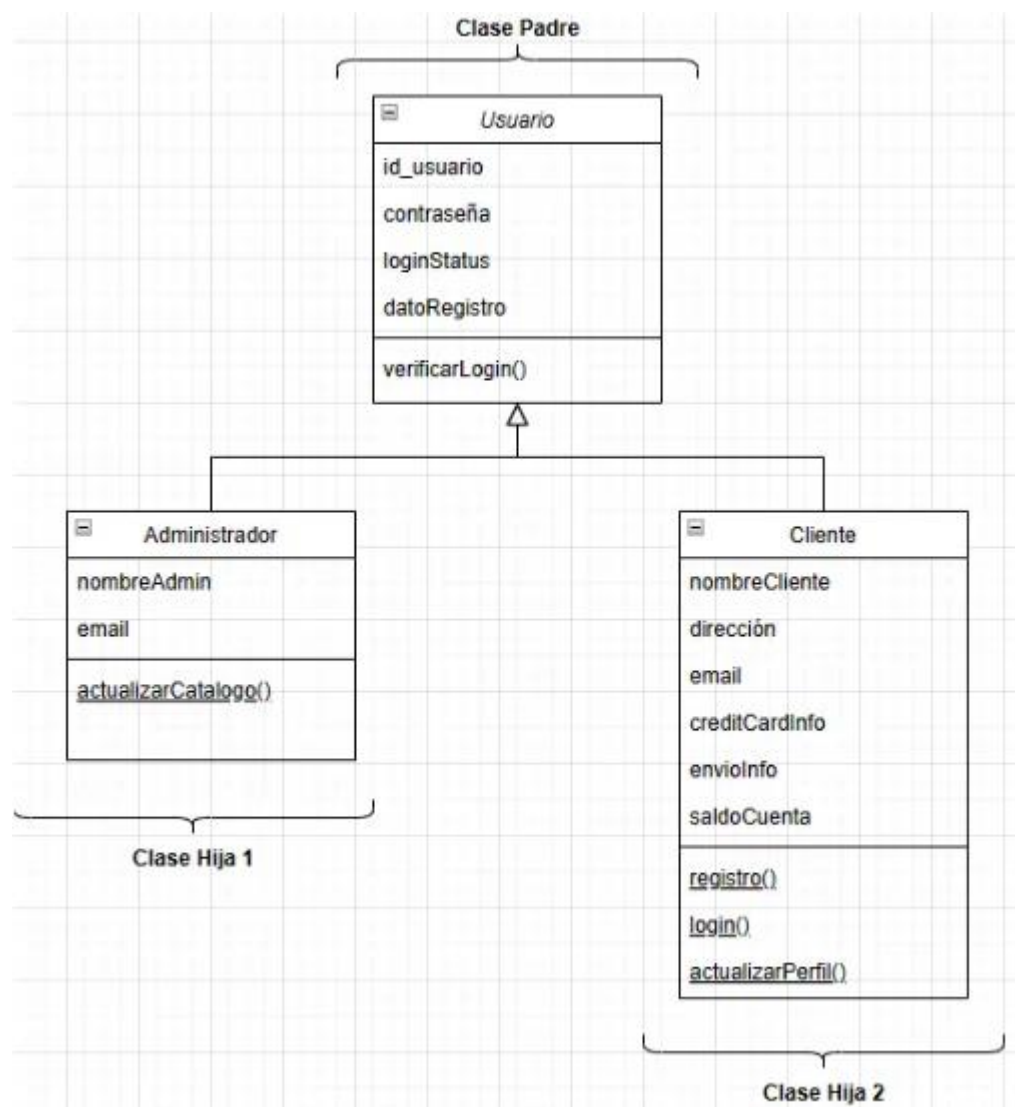
herramientas de revisión, características de administración de proyectos tal como asignación de tareas, estados de tareas, y más.

6. Crear dos clases Hijas y una clase Padre

A continuación, podemos observar en la Figura 1 un ejemplo de una clase padre y dos clases hijas.

Figura 10.

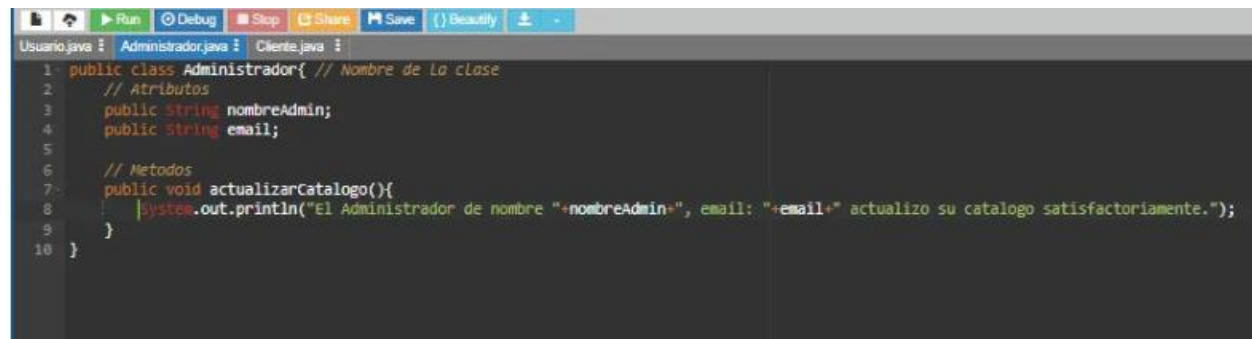
Ejemplo de clase Padre e Hija



9. Consulta No. 2

Crear 5 objetos con código y Diagramas UML.

Capturas del código JAVA



```
1 public class Administrador{ // Nombre de la clase
2     // Atributos
3     public String nombreAdmin;
4     public String email;
5
6     // Metodos
7     public void actualizarCatalogo(){
8         System.out.println("El Administrador de nombre "+nombreAdmin+", email: "+email+" actualizo su catalogo satisfactoriamente.");
9     }
10 }
```



```
1 public class Cliente{ // nombre de la clase
2     // Atributos
3     public String nombreCliente;
4     public String direccion;
5     public String email;
6     public float saldoCuenta;
7
8     // Metodos
9     public void registro(){
10         System.out.println("El Administrador de nombre "+nombreCliente+" con direccion "+direccion+", email: "+email+" con saldo de cuenta: "+saldoCuenta+" se registro satisfactoriamente.");
11     }
12 }
```

```

1  /*****
2  UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
3  DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION
4  ASIGNATURA: PROGRAMACION ORIENTADO A OBJETOS
5  DOCENTE: MGR LUIS ENRIQUE JARAMILLO MONTAÑO
6  ALUMNO: DAVID IVAN GRANADA PACHACAMA
7
8  Ejercicio: Crear 5 objetos
9  *****/
10 public class Usuario
11 {
12     public static void main(String[] args) {
13         //Creando primer Objeto
14         Administrador adm=new Administrador();
15         adm.nombreAdmin="David Granada";
16         adm.email="digranada@espe.edu.ec";
17         adm.actualizarCatalogo();
18         //Creando segundo Objeto
19         Administrador adm2=new Administrador();
20         adm2.nombreAdmin="Diana Lopez";
21         adm2.email="dilopez@espe.edu.ec";
22         adm2.actualizarCatalogo();
23         //Creando tercer Objeto
24         Administrador adm3=new Administrador();
25         adm3.nombreAdmin="Santiago Tenorio";
26         adm3.email="srtenorio@espe.edu.ec";
27         adm3.actualizarCatalogo();
28         //Creando cuarto Objeto
29         Cliente cln=new Cliente();
30         cln.nombreCliente="Maria Remache";
31         cln.direccion="Sangolqui";
32         cln.email="maremache@espe.edu.ec";
33         cln.saldoCuenta=780;
34         cln.registro();
35         //Creando quinto Objeto
36         Cliente cln2=new Cliente();
37         cln2.nombreCliente="Anita Hidalgo";
38         cln2.direccion="Quito";
39         cln2.email="ahidalgo@espe.edu.ec";
40         cln2.saldoCuenta=1025;
41         cln2.registro();
42     }
43 }

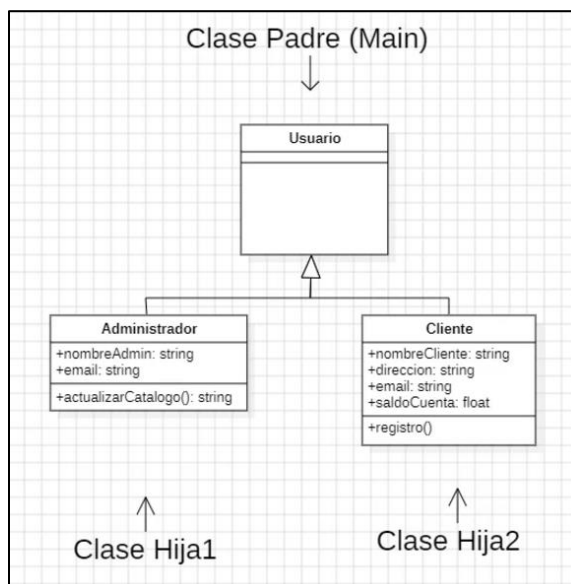
```

```

input
E1 Administrador de nombre David Granada, email: digranada@espe.edu.ec actualizo su catalogo satisfactoriamente.
E1 Administrador de nombre Diana Lopez, email: dilopez@espe.edu.ec actualizo su catalogo satisfactoriamente.
E1 Administrador de nombre Santiago Tenorio, email: srtenorio@espe.edu.ec actualizo su catalogo satisfactoriamente.
E1 Administrador de nombre Maria Remache con direccion Sangolqui, email: maremache@espe.edu.ec con saldo de cuenta: 780.0 se registro satisfactoriamente.
E1 Administrador de nombre Anita Hidalgo con direccion Quito, email: ahidalgo@espe.edu.ec con saldo de cuenta: 1025.0 se registro satisfactoriamente.
...Program finished with exit code 0

```

Diagramas de UML



10. Conclusiones

- Git debe instalarse en la última versión para que se pueda manejar los comandos mediante consola cmd o PowerShell.
- Al crear u repositorio en GitHub vía online podemos editar código en cualquier lenguaje de programación en donde queramos.
- En los Sistemas de control de versionamiento (VCS) podemos observar una mejora en la colaboración de equipo, ya que un VCS como Git permite a múltiples desarrolladores trabajar de manera simultánea en el mismo proyecto sin sobre escribir el trabajo de los demás.

11. Recomendaciones

- Utilizar ramas de manera efectiva: es recomendable usar ramas para organizar el trabajo en distintas funcionales o corrección de errores, esto permite que cada miembro del proyecto de manera aislada en tareas específicas sin interferir con el código principal, y al finalizar las ramas se pueden fusionar de manera controlada.
- Realizar Commits frecuentes y descriptivos: Asegurarse de que los mensajes sean claros y detallados, un buen mensaje de commit debe describir lo que ha hecho.
- Usar revisión de código y Pull Requests: Fomentar el uso de code review y PR antes de fusionar cambios a la rama principal, es esencial para mantener la calidad del código y evitar errores.

12. Referencias Bibliográficas

Borrell Nogueras, G. (2006). El control de versiones.

<https://torroja.dmt.upm.es/media/files/cversiones.pdf>

Inicio rápido para repositorios—Documentación de GitHub. (s. f.). GitHub Docs. Recuperado 20 de noviembre de 2024, de <https://docs.github.com/es/repositories/creating-and-managing-repositories/quickstart-for-repositories>

Git y GitHub—Aprende desarrollo web | MDN. (2024, julio 28).

https://developer.mozilla.org/es/docs/Learn/Tools_and_testing/GitH