



Análisis y diseños de sistemas

Docente: Juan Miranda.

Experiencia de Aprendizaje 2

Investigación sobre la utilización de Github

Integrantes:

Cea Laínez	Hugo Alejandro	CL222456
Ramos Fernández	Néstor Uriel	RF221538

Ciudadela Don Bosco, 26 de octubre del 2023

Contenido

Objetivos	3
Introducción:.....	3
Marco Teórico	3
Qué es Git.....	3
Objetos de Git.....	3
Qué es Github.....	4
¿Qué es una metodología de desarrollo de software en el mundo de la programación y de qué forma se puede aplicar?	4
Git&Github, Setup y Config	4
Setup clave SSH.....	5
Clonar el repositorio en tu sistema local:.....	6
Realizar cambios, revertirlos y subirlos al repositorio remoto.	6
Ejemplos de su utilización, con imágenes ilustrativas	7
Crear un repositorio de ejemplo con cualquier tipo de contenido orientado al desarrollo web	11
Conclusiones.....	11
Referencias:	12

Objetivos

General: Promover una comprensión profunda y práctica del uso de GitHub en el desarrollo de software, destacando su importancia, beneficios y mejores prácticas, con el fin de capacitar a los desarrolladores y equipos para maximizar su eficiencia

Específicos:

1. Comprender el uso de Github en el proceso de programación.
2. Mostrar los diversos procesos a los que podemos tener accesos gracias a Github.
3. Ilustrar mediante ejemplos el uso de Github.

Introducción:

El presente documento abordará el uso de GitHub, una plataforma líder en el control de versiones y la colaboración en desarrollo de software, ha revolucionado la forma en que los equipos trabajan juntos, gestionan proyectos y comparten código. Este documento explora el marco teórico del uso de GitHub, desde los principios fundamentales de control de versiones hasta su integración con metodologías ágiles y DevOps, destacando su papel esencial en la comunidad de desarrollo de software.

Marco Teórico

Qué es Git

Git es un sistema de control de versiones distribuido ampliamente, utilizado para el seguimiento de cambios en archivos y colaboración en proyectos de desarrollo de software. Fue creado por Linus Torvalds en 2005 y se ha convertido en una herramienta fundamental en el desarrollo de software, y en una parte esencial de la mayoría de los flujos de trabajo de programadores y equipos de desarrollo.

Objetos de Git

Los objetos fundamentales en Git son las unidades de datos que Git almacena internamente para gestionar y rastrear el historial de versiones de tus proyectos. Los cuatro objetos principales en Git son:

1. Blob: El objeto "blob" (binario large object) representa el contenido de un archivo en un punto específico de la historia. Contiene los datos binarios del archivo, y cada versión de un archivo en el repositorio se almacena como un objeto "blob" independiente.
2. Tree: El objeto "tree" representa un directorio en un punto específico del historial. Contiene una lista de archivos y subdirectorios, cada uno de los cuales puede estar asociado a un objeto "blob" o "tree" correspondiente.
3. Commit: El objeto "commit" es una instantánea de un conjunto de cambios en el proyecto en un momento específico. Contiene un puntero al objeto "tree" que representa el estado del

proyecto en ese commit, además de metadatos como el autor, la fecha, el mensaje del commit y un puntero a uno o varios commits padres (en el caso de fusiones).

4. Tag: El objeto "tag" (etiqueta) se utiliza para marcar puntos específicos en la historia del proyecto. Pueden ser utilizados para crear versiones estables, realizar lanzamientos o señalar hitos importantes en el desarrollo.

Estos objetos se combinan y enlazan para formar el historial de versiones de un proyecto en Git. Git gestiona estos objetos de manera eficiente y permite a los desarrolladores rastrear, fusionar, ramificar y gestionar cambios de manera efectiva en proyectos colaborativos.

En Git, los objetos se almacenan en un directorio llamado `.git` en la raíz del repositorio. Los usuarios no necesitan interactuar directamente con estos objetos, ya que Git proporciona comandos y herramientas para gestionarlos de manera transparente.

Qué es Github

GitHub es una plataforma de desarrollo colaborativo que aloja proyectos en la nube utilizando el sistema de control de versiones llamado Git. Ayuda a los desarrolladores a almacenar y administrar el código llevando un registro de cambios. Generalmente el código es abierto, lo que permite realizar proyectos compartidos y mantener el seguimiento detallado de su progreso.

¿Qué es una metodología de desarrollo de software en el mundo de la programación y de qué forma se puede aplicar?

Una metodología de desarrollo de software es un conjunto de prácticas, procesos y directrices que se utilizan para planificar, diseñar, construir, probar y entregar software de manera efectiva y eficiente. Estas metodologías proporcionan un enfoque estructurado para el desarrollo de software y se utilizan para gestionar el ciclo de vida de un proyecto de desarrollo. Hay muchas metodologías diferentes en el mundo de la programación, y cada una tiene sus propias características y enfoques específicos.

En Github se utiliza en conjunción con diversas metodologías y prácticas, como las metodologías ágiles y DevOps, para mejorar la colaboración, la gestión de proyectos, la revisión de código, la automatización de pruebas y el despliegue continuo en el desarrollo de software. GitHub es una herramienta esencial para equipos de desarrollo que desean trabajar de manera efectiva y eficiente en proyectos de software colaborativos.

Git&Github, Setup y Config

<https://docs.github.com/es/desktop/overview/getting-started-with-github-desktop>

Configurar Git y GitHub implica configurar Git en el sistema local y establecer una conexión con tu cuenta de GitHub.

Configuración de Git en el sistema local:

1. Instalar Git en el sistema local.
2. Configurar el nombre y dirección de correo electrónico abriendo una terminal o línea de comandos en donde se ejecutará:
git config --global user.name "nombre"
git config --global user.email "email"

Configuración de GitHub:

1. Crear una cuenta en github

Conexión entre Git y GitHub:

1. Configurar el nombre de usuario de GitHub y la dirección de correo electrónico (Se debe asegurar que el nombre de usuario y la dirección de correo electrónico en la configuración de Git coincidan con la cuenta en GitHub).

Setup clave SSH

Verificar la existencia de claves SSH:

Antes de generar una nueva clave SSH, se debe verificar si se tiene alguna clave SSH en el sistema. Abriendo una terminal o línea de comandos y ejecutar el siguiente comando: **ls -al ~/.ssh**

Si se ven archivos llamados id_rsa y id_rsa.pub, ya se tiene una clave SSH. Se puede usar la clave o generar una nueva según se quiera.

Generar una nueva clave SSH (si es necesario):

Si no se tiene una clave SSH o se desea generar una nueva, se utiliza el siguiente comando: **ssh-keygen -t rsa -b 4096 -C "email"** (el email debe ser el de la cuenta de github).

‘-t’ especifica el tipo de clave (RSA en este caso).

‘-b’ establece la longitud de la clave (4096 bits es una opción segura).

‘-C’ permite agregar un comentario (generalmente la dirección de correo).

El comando preguntará dónde se desea guardar la clave. Se puede presionar "Enter" para aceptar la ubicación predeterminada (/home/tu_usuario/.ssh/id_rsa) o especificar una ubicación diferente.

Establecer una contraseña (opcional):

Puedes configurar una contraseña adicional para proteger tu clave SSH. Esto proporciona una capa adicional de seguridad. Para hacerlo, ejecuta el siguiente comando: **ssh-keygen -p**.

Copiar la clave pública SSH:

Ahora se debe copiar la clave pública SSH al portapapeles para agregarla a la cuenta de GitHub. Se utiliza el siguiente comando para copiarla al portapapeles: **cat ~/.ssh/id_rsa.pub**

Agregar la clave SSH a la cuenta de GitHub:

1. Ir a GitHub en un navegador web.
2. Hacer clic en la foto de perfil en la esquina superior derecha y seleccionar "Settings".
3. En el menú lateral izquierdo, elegir "SSH and GPG keys" (Claves SSH y GPG).
4. Hacer clic en "New SSH key" (Nueva clave SSH).
5. En el campo "Title" (Título), proporcionar un nombre descriptivo para la clave.
6. En el campo "Key" (Clave), pega la clave SSH que se copió en el Paso 4.
7. Hacer clic en "Add SSH key" (Agregar clave SSH).

8) Crear y clonar un repositorio:

Crear un nuevo repositorio en GitHub:

1. Iniciar sesión en la cuenta de GitHub.
2. En la esquina superior derecha de la página, hacer clic en el botón "▼" junto a la foto de perfil y seleccionar "Your repositories" (Tus repositorios).
3. En la página "Your repositories" hacer clic en el botón "New" (Nuevo) para crear un nuevo repositorio.
4. Completar la información requerida para el nuevo repositorio, incluyendo el nombre del repositorio, una descripción opcional, visibilidad (público o privado), opciones de inicialización y otras configuraciones.
5. Hacer clic en el botón "Create repository" (Crear repositorio) para crear el nuevo repositorio en GitHub.

Clonar el repositorio en tu sistema local:

Una vez que se haya creado el repositorio en GitHub, se puede clonar en el sistema local utilizando Git. Abriendo una terminal o línea de comandos y ejecutando el siguiente comando: **git clone https://github.com/tu-usuario/nombre-del-repositorio.git**.

Asegurarse de que el repositorio sea público o que se tenga permisos de acceso para clonarlo.

Realizar cambios, revertirlos y subirlos al repositorio remoto.

Realizar cambios en el repositorio local:

1. Antes de hacer un commit, verifica el estado de los cambios en tu repositorio local utilizando el siguiente comando: **git status**. Esto mostrará los archivos modificados,

los archivos nuevos, los archivos eliminados y otros detalles sobre el estado del repositorio.

2. Hacer un commit de los cambios, agregar los archivos que se desean incluir en el commit utilizando el comando git add. Por ejemplo, para agregar todos los cambios, se puede usar: **git add .**
3. realiza un commit de los cambios con un mensaje descriptivo: **git commit -m "Mensaje descriptivo de los cambios"**

Revertir los cambios:

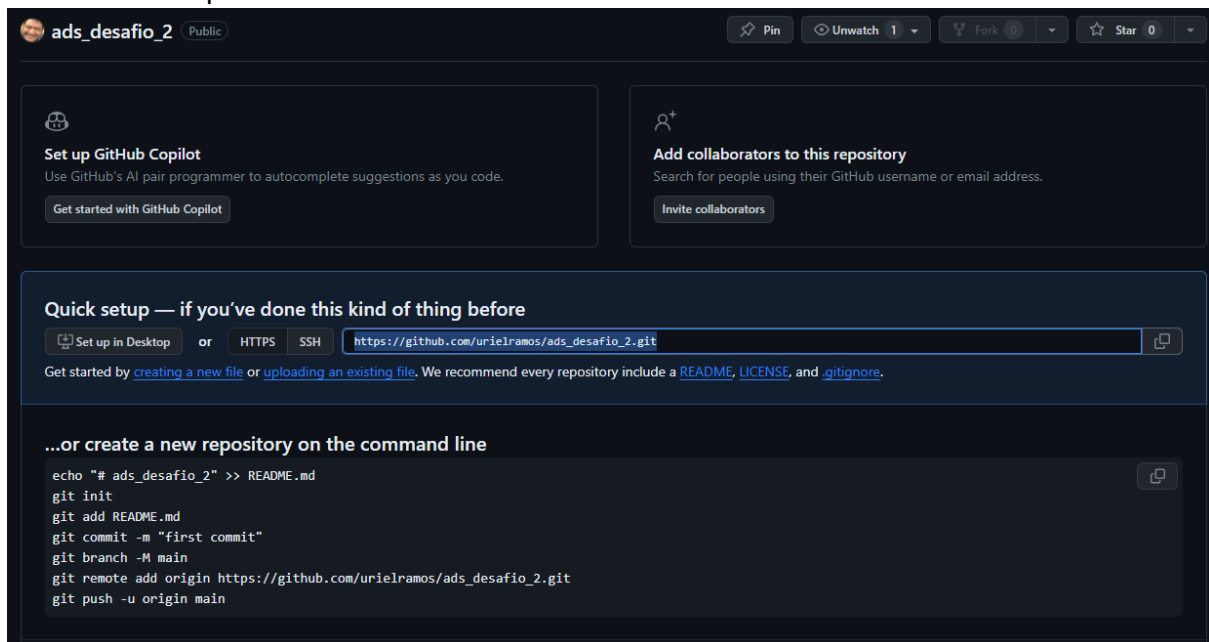
Si se desea revertir los cambios que se realizaron en un archivo específico o un conjunto de archivos, se puede usar el comando git checkout: **git checkout nombre-del-archivo**

Subir los cambios al repositorio remoto:

1. Una vez que se hayan realizado los cambios y confirmado el commit, se pueden subir los cambios al repositorio remoto en GitHub. Utilizando el comando git push: **git push origin nombre-de-la-rama**
2. Reemplazar nombre-de-la-rama con el nombre de la rama en la que se desea subir los cambios. Si se está trabajando en la rama principal (generalmente llamada "master" o "main"), se puede simplemente usar: **git push**

Ejemplos de su utilización, con imágenes ilustrativas

Creación del repositorio Github sobre la versión del documento de la tarea



https://github.com/urielramos/ads_desafio_2

Proceso de clonación del repositorio y carga inicial del documento

```
MINGW64/c:/Users/nramos/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Dise
ño de Sistemas ADS901 G01T/desafío 2
$ git clone https://github.com/urielramos/ads_desafio_2.git
Cloning into 'ads_desafio_2'...
warning: You appear to have cloned an empty repository.

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Dise
ño de Sistemas ADS901 G01T/desafío 2
$ cd ads_desafio_2

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Dise
ño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    desafio 2.docx

nothing added to commit but untracked files present (use "git add" to track)

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Dise
ño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git add .

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Dise
ño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git commit -m "Archivo inicial de tarea"
[main (root-commit) eb287ce] Archivo inicial de tarea
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 desafio 2.docx

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Dise
ño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git push
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 18.66 KiB | 18.66 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/urielramos/ads_desafio_2.git
 * [new branch]      main -> main

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Dise
ño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$
```

Primera versión de la tarea publicada

The screenshot shows the GitHub interface for the repository 'ads_desafio_2' by user 'urielramos'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a single commit 'eb287ce' titled 'Archivo inicial de tarea' made 2 minutes ago. The file list shows 'desafio 2.docx' as the only file. The repository has 0 stars, 1 watcher, and 0 forks. There is a prompt to 'Add a README' to help people understand the project. The 'About' section is empty, and the 'Releases' section shows no published releases.

Se agrega portada al documento y se actualiza en el repositorio

```
MINGW64/c/Users/nramos/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  ~$safio 2.docx

nothing added to commit but untracked files present (use "git add" to track)

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git add .

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git commit -m "Se agrega la portada de documento"
[main 4b33c51] Se agrega la portada de documento
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ~$safio 2.docx

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 412 bytes | 412.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/urielramos/ads_desafio_2.git
eb287ce..4b33c51 main -> main

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$
```

Se cambia el nombre de documento para facilitar su actualización. Se elimina archivo temporal.

```
MINGW64/c/Users/nramos/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    desafio 2.docx
    deleted:    ~$safio 2.docx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  desafio2.docx

no changes added to commit (use "git add" and/or "git commit -a")

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git add .

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git commit -m "Fix: se cambia nombre del documento para facilitar su actualización al repositorio"
[main 232b4f4] Fix: se cambia nombre del documento para facilitar su actualización al repositorio
3 files changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 desafio 2.docx
create mode 100644 desafio2.docx
delete mode 100644 ~$safio 2.docx

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 334.87 KiB | 25.76 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/urielramos/ads_desafio_2.git
4b33c51..232b4f4 main -> main

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafío 2/ads_desafio_2 (main)
$ |
```

Se ordena el documento, se agregan títulos y subtítulos. Se agregan imágenes de ejemplo del uso de un repositorio.

```
MINGW64/c:/Users/nramos/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   desafio2.docx

no changes added to commit (use "git add" and/or "git commit -a")

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git add .

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git commit -m "feat: se ordena el contenido del documento y se crean los títulos y subtítulos. Se agregan imágenes de ejemplo del uso del repositorio"
[main 07a665b] feat: se ordena el contenido del documento y se crean los títulos y subtítulos. Se agregan imágenes de ejemplo del uso del repositorio
1 file changed, 0 insertions(+), 0 deletions(-)

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 86.46 KiB | 9.61 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/urielramos/ads_desafio_2.git
   232b4f4..07a665b  main -> main

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ |
```

Se agrega imagen de creación de repositorio y conclusiones

```
MINGW64/c:/Users/nramos/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   desafio2.docx

no changes added to commit (use "git add" and/or "git commit -a")

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git add .

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git commit -m "feat: se agrega imagen de creación de un repositorio y conclusiones"
[main 78808a9] feat: se agrega imagen de creación de un repositorio y conclusiones
1 file changed, 0 insertions(+), 0 deletions(-)

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 116.73 KiB | 9.73 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/urielramos/ads_desafio_2.git
   07a665b..78808a9  main -> main

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ |
```

Se agrega al documento el índice de contenido

```
MINGW64/c:/Users/nramos/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2
nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   desafio2.docx

no changes added to commit (use "git add" and/or "git commit -a")

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git add .

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git commit -m "feat: se agrega indice del contenido"
[main 8029bac] feat: se agrega indice del contenido
1 file changed, 0 insertions(+), 0 deletions(-)

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 83.94 KiB | 6.99 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/urielramos/ads_desafio_2.git
   78808a9..8029bac  main -> main

nramos@I010LAYL33 MINGW64 ~/Documents/proyectos/otros/udb/202302/Análisis y Diseño de Sistemas ADS901 G01T/desafio 2/ads_desafio_2 (main)
$ |
```


Crear un repositorio de ejemplo con cualquier tipo de contenido orientado al desarrollo web

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 *

 **urielramos**


Repository name *

ads_desafio_I|

 **ads_desafio_II is available.**

Great repository names are short and memorable. Need inspiration? How about **fictional-parakeet** ?

Description (optional)

☒  **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.

Conclusiones

Git es una herramienta esencial en el desarrollo de software debido a sus ventajas en el control de versiones, colaboración en equipo, flexibilidad y rendimiento. Su uso efectivo puede mejorar la productividad de los desarrolladores y la calidad del software producido.

Referencias:

<https://ebac.mx/blog/que-es-github>

<https://www.ibm.com/docs/es/developer-for-zos/15.0?topic=guide-concepts>

<https://docs.github.com/es/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>