**Universidad Don Bosco**

##### **Técnico en Ingeniería en Computación**



**Asignatura:**

Desarrollo de Aplicaciones Web con Software Interpretados en el Cliente

**Actividad:**

Git y GitHub

**Integrantes:**

Lorena Isabel Nájera Urbina NU222221

José Mario Ayala Reyna AR120420

José Alejandro Hernández Ramírez HR211569

**Docente:**

Ing. Joshua Reyes

21 de noviembre de 2022

**Índice**

***[Introducción 3](#_heading=h.gjdgxs)***

***[Objetivo 3](#_heading=h.30j0zll)***

***[Objetivos especifico 3](#_heading=h.1fob9te)***

***[¿Qué es Git? 4](#_heading=h.3znysh7)***

***[Conceptos básicos de Git 4](#_heading=h.2et92p0)***

**[Ramas 4](#_heading=h.tyjcwt)**

**[Archivos y confirmaciones 5](#_heading=h.3dy6vkm)**

***[Ventajas de Git 5](#_heading=h.1t3h5sf)***

**[Desarrollo simultáneo 5](#_heading=h.4d34og8)**

**[Versiones más rápidas 5](#_heading=h.2s8eyo1)**

**[Integración integrada 5](#_heading=h.17dp8vu)**

**[Git funciona con cualquier equipo 6](#_heading=h.3rdcrjn)**

***[¿Qué es GitHub? 7](#_heading=h.26in1rg)***

***[Crear ramas Github 7](#_heading=h.lnxbz9)***

***[Entender los commits de GitHub 7](#_heading=h.35nkun2)***

***[Crear solicitudes de extracción en GitHub 8](#_heading=h.1ksv4uv)***

***[Flujograma de nuestra solución 9](#_heading=h.44sinio)***

***[Conclusión 10](#_heading=h.2jxsxqh)***

# Introducción

La tecnología está en constante cambio, esta evoluciona día con día y es importante actualizarse e informarse para darle el uso adecuado y adaptarlas a nuestro entorno.

En la práctica, es importante apoyarse de herramientas que facilitarán nuestro trabajo, por lo tanto, con esta investigación nos informaremos sobre los controladores de versiones que podemos utilizar para nuestros proyectos de desarrollo.

# Objetivo

Profundizar sobre las herramientas de control de versiones mediante la investigación.

# Objetivos especifico

1. Definir la funcionalidad Git y GitHub
2. Conocer las características de las herramientas Git y GitHub.



# ¿Qué es Git?

Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, se ha convertido en el estándar mundial para el control de versiones.

o que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.

# Conceptos básicos de Git

Cada vez que se guarda el trabajo, Git crea una confirmación. Una confirmación es una instantánea de todos los archivos en un momento dado. Si un archivo no ha cambiado de una confirmación a la siguiente, Git usa el archivo almacenado anteriormente. Este diseño difiere de otros sistemas que almacenan una versión inicial de un archivo y mantienen un registro de deltas a lo largo del tiempo.

Las confirmaciones crean vínculos a otras confirmaciones, formando un gráfico del historial de desarrollo. Es posible revertir el código a una confirmación anterior, inspeccionar cómo cambiaron los archivos de una confirmación a la siguiente y revisar información como dónde y cuándo se realizaron los cambios. Las confirmaciones se identifican en Git mediante un hash criptográfico único del contenido de la confirmación. Dado que todo está hash, es imposible realizar cambios, perder información o archivos dañados sin que Git lo detecte.

## Ramas

Cada desarrollador guarda los cambios en su propio repositorio de código local. Como resultado, puede haber muchos cambios diferentes en función de la misma confirmación. Git proporciona herramientas para aislar los cambios y volver a combinarlos posteriormente. Las ramas, que son punteros ligeros para trabajar en curso, administran esta separación. Una vez finalizado el trabajo creado en una rama, se puede combinar de nuevo en la rama principal (o troncal) del equipo.

## Archivos y confirmaciones

Los archivos de Git se encuentran en uno de los tres estados: modificados, almacenados provisionalmente o confirmados. Cuando se modifica un archivo por primera vez, los cambios solo existen en el directorio de trabajo. Todavía no forman parte de una confirmación o del historial de desarrollo. El desarrollador debe almacenar provisionalmente los archivos modificados que se incluirán en la confirmación. El área de almacenamiento provisional contiene todos los cambios que se van a incluir en la siguiente confirmación. Una vez que el desarrollador esté satisfecho con los archivos almacenados provisionalmente, los archivos se empaquetan como confirmación con un mensaje que describe lo que ha cambiado. Esta confirmación forma parte del historial de desarrollo.

# Ventajas de Git

## Desarrollo simultáneo

Todos tienen su propia copia local de código y pueden trabajar simultáneamente en sus propias ramas. Git funciona sin conexión, ya que casi todas las operaciones son locales.

## Versiones más rápidas

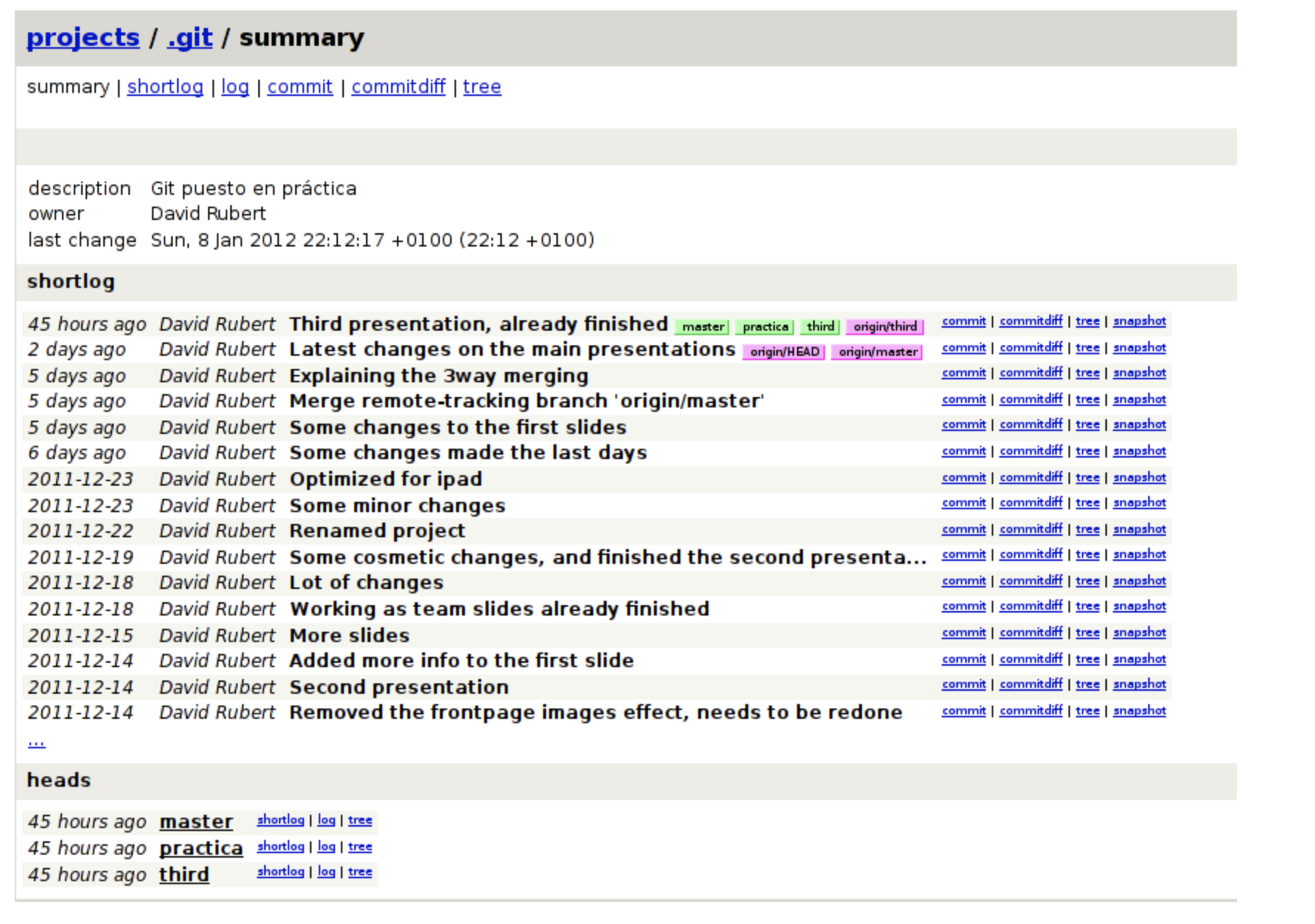
Las ramas permiten el desarrollo flexible y simultáneo. La rama principal contiene código estable y de alta calidad desde el que se publica. Las ramas de características contienen trabajo en curso, que se combinan en la rama principal tras la finalización. Al separar la rama de versión del desarrollo en curso, es más fácil administrar código estable y enviar actualizaciones más rápidamente.

## Integración integrada

Debido a su popularidad, Git se integra en la mayoría de las herramientas y productos. Cada IDE principal tiene compatibilidad integrada con Git y muchas herramientas admiten la integración continua, la implementación continua, las pruebas automatizadas, el seguimiento de elementos de trabajo, las métricas y la integración de características de informes con Git. Esta integración simplifica el flujo de trabajo diario.

## Git funciona con cualquier equipo

El uso de Git con una herramienta de administración de código fuente aumenta la productividad de un equipo fomentando la colaboración, aplicando directivas, automatizando procesos y mejorando la visibilidad y la rastreabilidad del trabajo. El equipo puede establecerse en herramientas individuales para el control de versiones, el seguimiento de elementos de trabajo y la integración e implementación continuas. O bien, pueden elegir una solución como GitHub o Azure DevOps que admita todas estas tareas en un solo lugar.



# ¿Qué es GitHub?



Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft [en junio del 2018](https://www.xataka.com/aplicaciones/oficial-microsoft-compra-github-7-500-millones-dolares). La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

Git es uno de estos sistemas de control, que permite comparar el código de un archivo para ver las diferencias entre las versiones, restaurar versiones antiguas si algo sale mal, y fusionar los cambios de distintas versiones. También permite trabajar con distintas ramas de un proyecto, como la de desarrollo para meter nuevas funciones al programa o la de producción para depurar los bugs.

Las principales características de la plataforma es que ofrece las mejores características de este tipo de servicios sin perder la simplicidad, y es una de las más utilizadas del mundo por los desarrolladores. Es multiplataforma, y tiene multitud de interfaces de usuario.

# Crear ramas Github

Con la creación de ramas, se generan diferentes versiones de un repositorio. Al hacer cambios en el proyecto en la rama de características, un desarrollador puede ver cómo afectará al proyecto maestro cuando se integre.

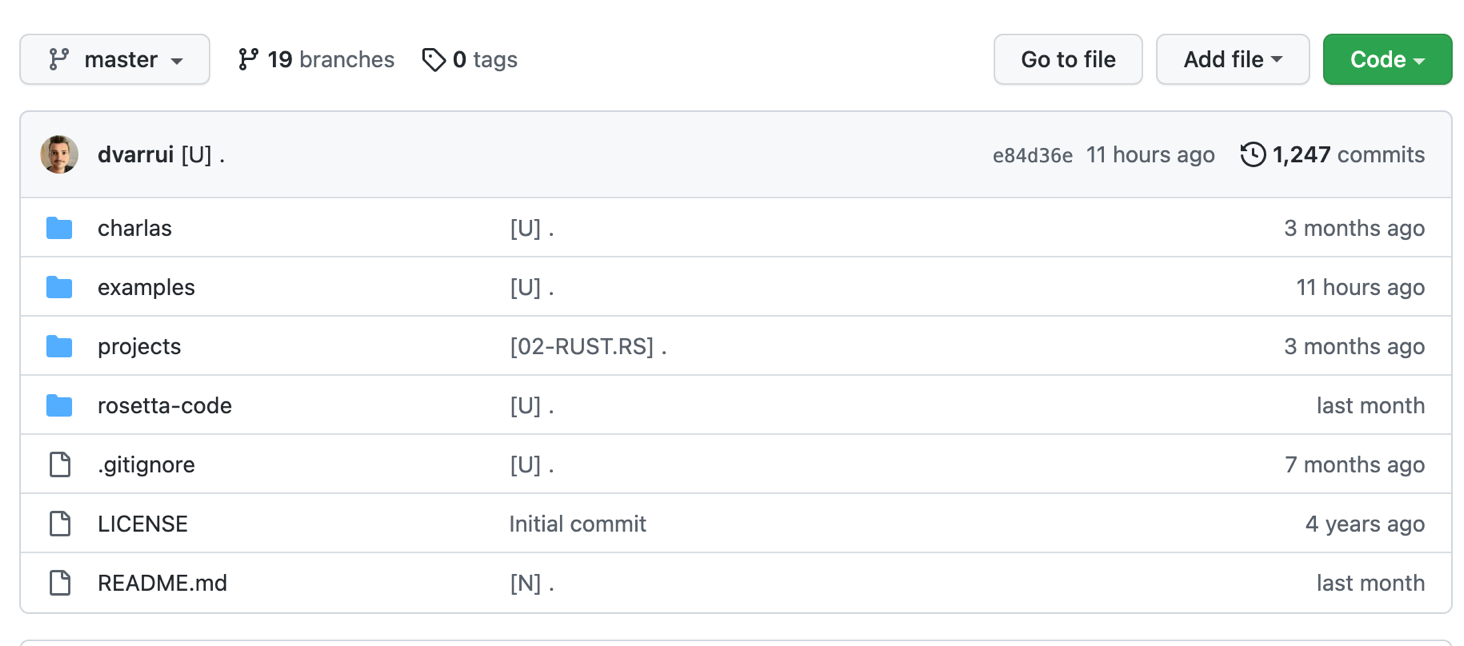
# Entender los commits de GitHub

Los commits son la forma en que se denominan los cambios guardados en GitHub. Cada vez que se cambie el archivo de la rama de características, se debe hacer un Commit para mantenerlo.

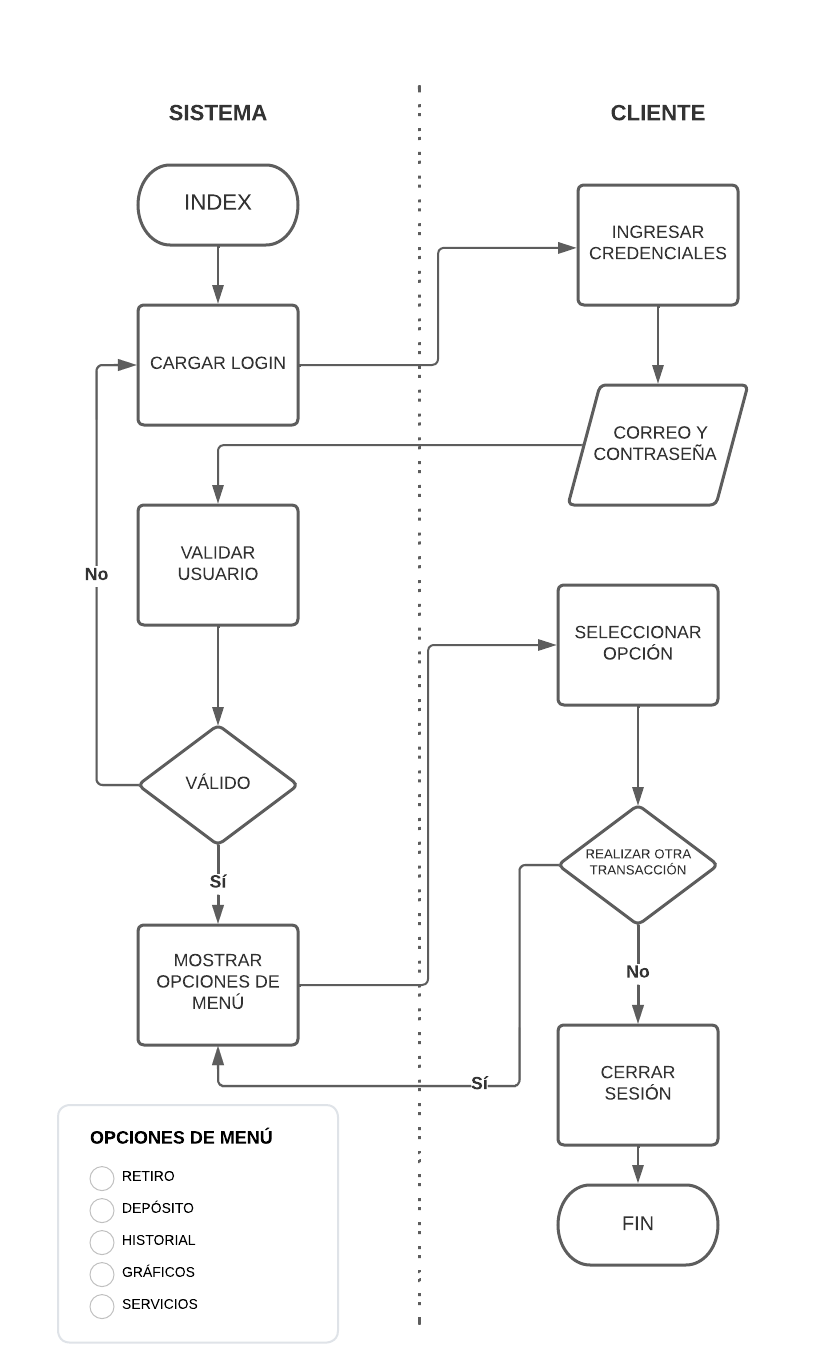
# Crear solicitudes de extracción en GitHub

Para proponer los cambios que se acaban de hacer a otros desarrolladores que trabajan en el mismo proyecto, se debe crear una solicitud de extracción. Estas facilitan el trabajo conjunto en los proyectos, ya que son la principal herramienta de colaboración en GitHub.

Las solicitudes de extracción te permiten ver las diferencias entre el proyecto original y tu rama de características. Es la forma de pedir a tus compañeros que las revisen. Si los otros desarrolladores lo aprueban, pueden fusionar la solicitud de extracción, lo que aplicará esos cambios al proyecto principal.



# Flujograma de nuestra solución



# Conclusión

Con esta investigación, hemos aprendido sobre las herramientas que nos facilitan el control de versiones de nuestros desarrollos, que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo y ayuda a los equipos a resolver este tipo de problemas al realizar un seguimiento de todos los cambios individuales de cada colaborador y al contribuir a evitar que el trabajo concurrente entre en conflicto.