DOCUMENTACIÓN

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

**SISTEMAS DE CONTROL DE VERSIONES O VCS**

Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico). Ejemplos de este tipo de herramientas son entre otros:

* [CVS](https://es.wikipedia.org/wiki/CVS), [Subversion](https://es.wikipedia.org/wiki/Subversion), [SourceSafe](https://es.wikipedia.org/wiki/SourceSafe), [ClearCase](https://en.wikipedia.org/wiki/Rational_ClearCase), [Darcs](https://es.wikipedia.org/wiki/Darcs), [Bazaar](https://es.wikipedia.org/wiki/Bazaar_(software)), [Plastic SCM](https://es.wikipedia.org/wiki/Plastic_SCM), [Git](https://es.wikipedia.org/wiki/Git), [Mercurial](https://es.wikipedia.org/wiki/Mercurial), [Perforce](https://es.wikipedia.org/wiki/Perforce), [Fossil SCM](https://es.wikipedia.org/wiki/Fossil_(gesti%C3%B3n_de_configuraci%C3%B3n_de_software)), Team Foundation Server.

**SISTEMAS DE CONTROL DE VERSIONES CENTRALIZADOS**

El siguiente gran problema que se encuentra la gente es que necesitan colaborar con desarrolladores en otros sistemas.

Para solventar este problema, se desarrollaron los sistemas de control de versiones centralizados (Centralized Version Control Systems o CVCSs en inglés). Estos sistemas, como CVS, Subversion, y Perforce, tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos desde ese lugar central. Durante muchos años éste ha sido el estándar para el control de versiones.

**SISTEMAS DE CONTROL DE VERSIONES DISTRIBUIDOS**

Es aquí donde entran los sistemas de control de versiones distribuidos (Distributed Version Control Systems o DVCSs en inglés). En un DVCS (como Git, Mercurial, Bazaar o Darcs), los clientes no sólo descargan la última instantánea de los archivos: replican completamente el repositorio. Así, si un servidor muere, y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo. Cada vez que se descarga una instantánea, en realidad se hace una copia de seguridad completa de todos los datos

* **GITHUB** (Trabaja de manera gráfica)
* GIT (Utiliza la línea de consola)

Es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones [Git](http://es.wikipedia.org/wiki/Git). Administrar código fuente.

* Crear una cuenta (Correo valido y contraseña)
* Iniciar sesión

**¿Para qué sirve?**

GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

Además de eso, puedes contribuir a mejorar el software de los demás. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls.

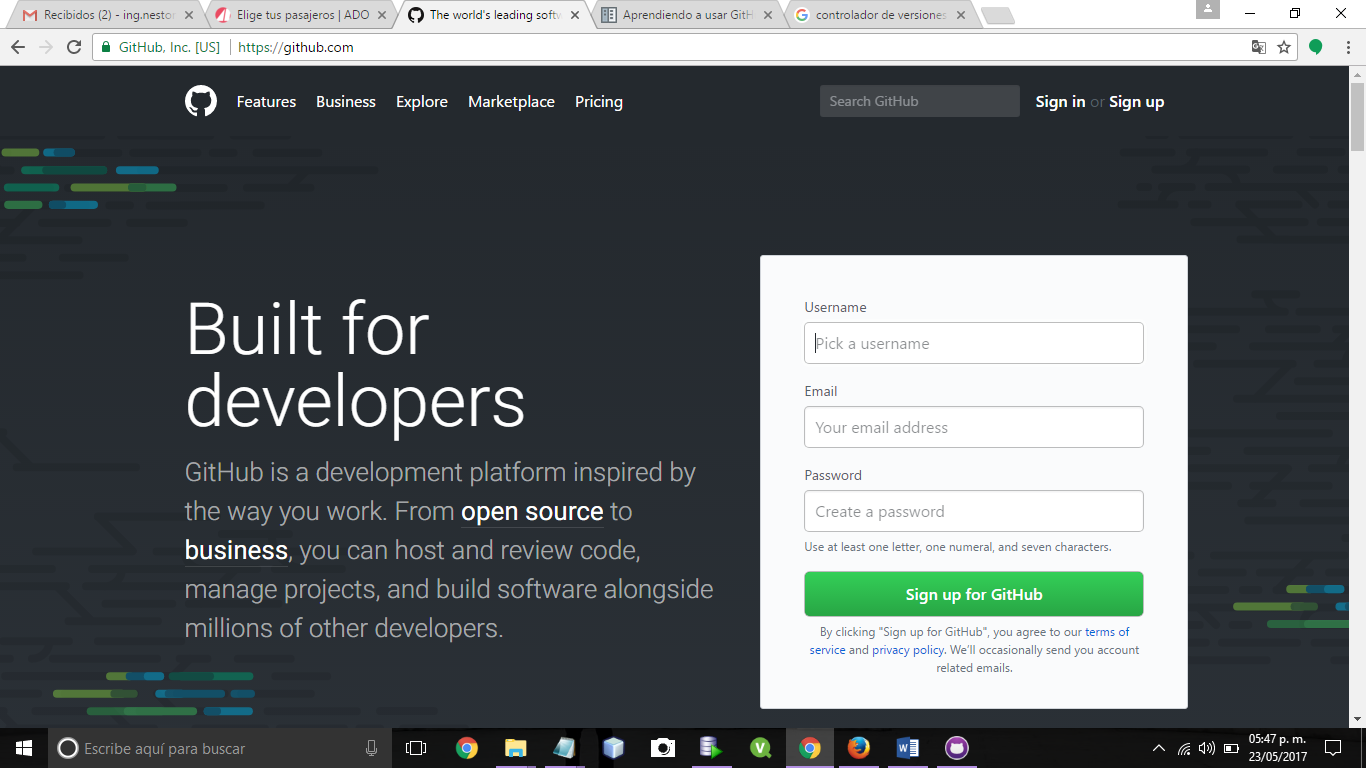
Realizar un fork es simplemente clonar un repositorio ajeno (genera una copia en tu cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas tus modificaciones puedes enviar un pull al dueño del proyecto.

Éste podrá analizar los cambios que has realizado fácilmente, y si considera interesante tu contribución, adjuntarlo con el repositorio original.

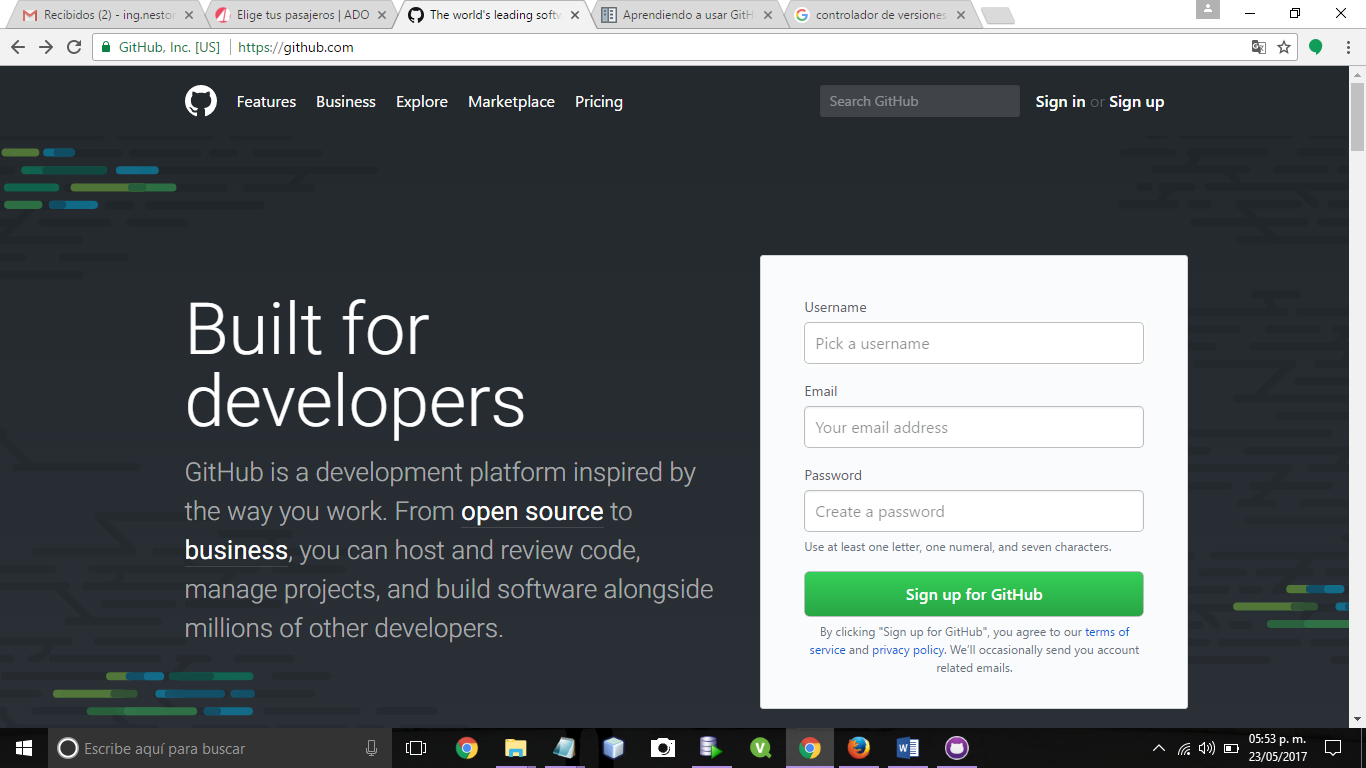
1.- Crear una Cuenta

Para crear una cuenta GitHub vamos a seguir los siguientes pasos:

* Entra a [GitHub](https://www.github.com/) e ingresar los datos que te piden:
* Nombre de usuario.
* Dirección email.
* Contraseña.

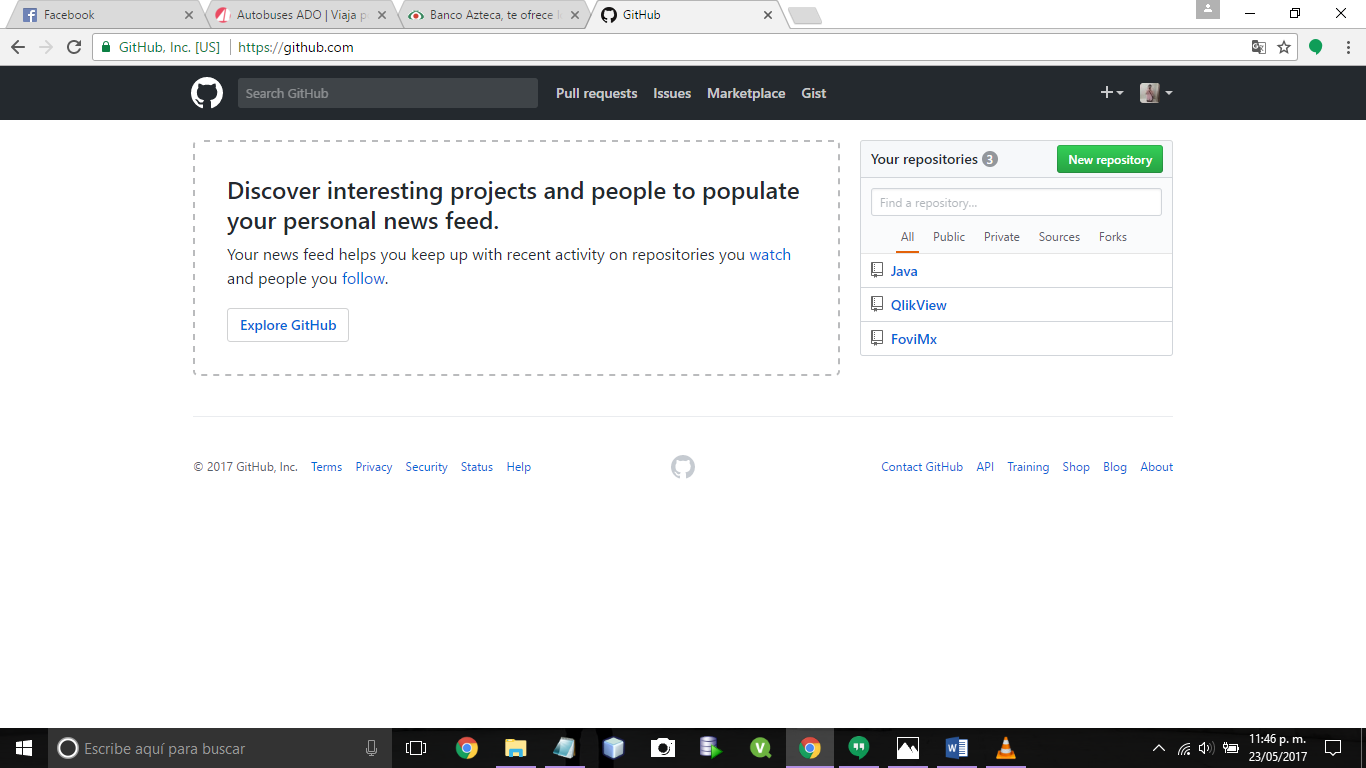


2.- Si ya cuentas con una cuenta, ingresar en el apartado de Sign in

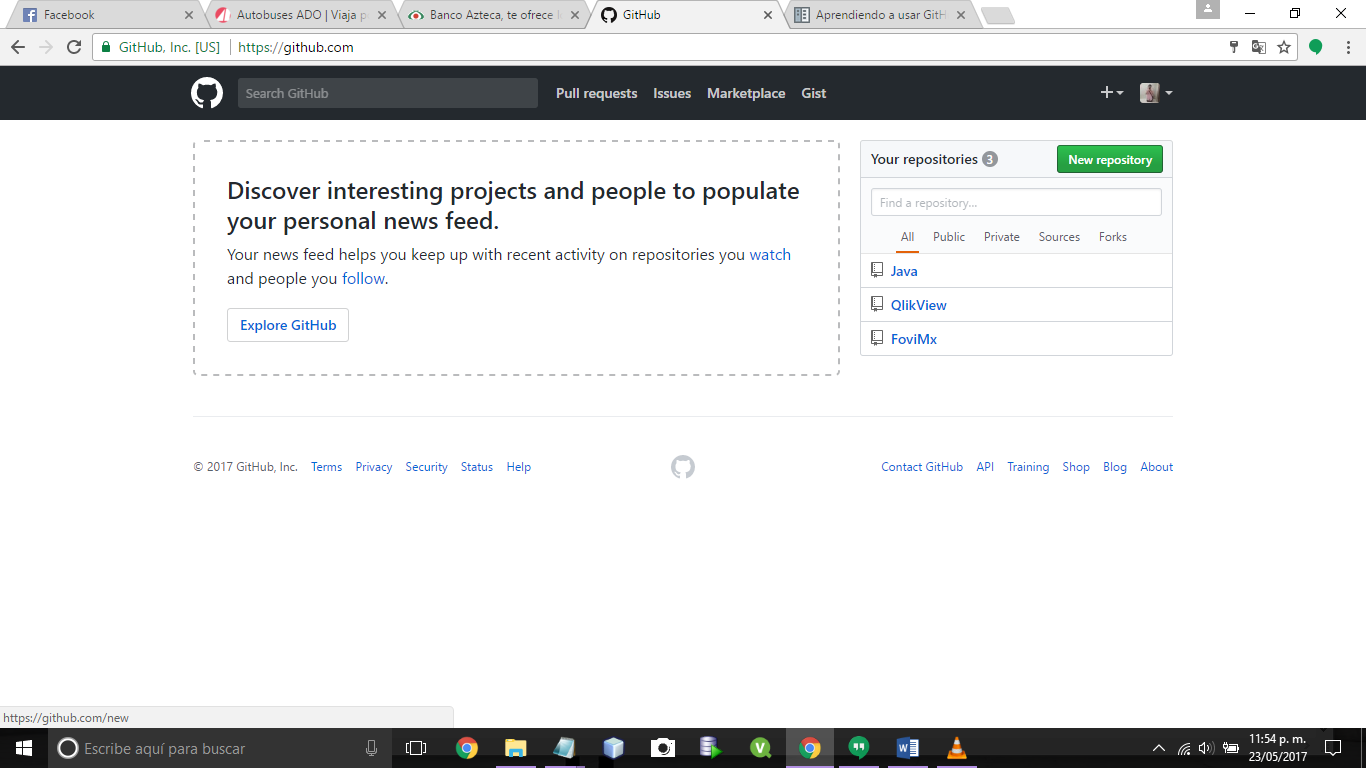


Iniciar Sesión

3.- Al ingresar con la cuenta registrada, mostrara la siguiente interfaz, en el cual procederemos a crear un repositorio

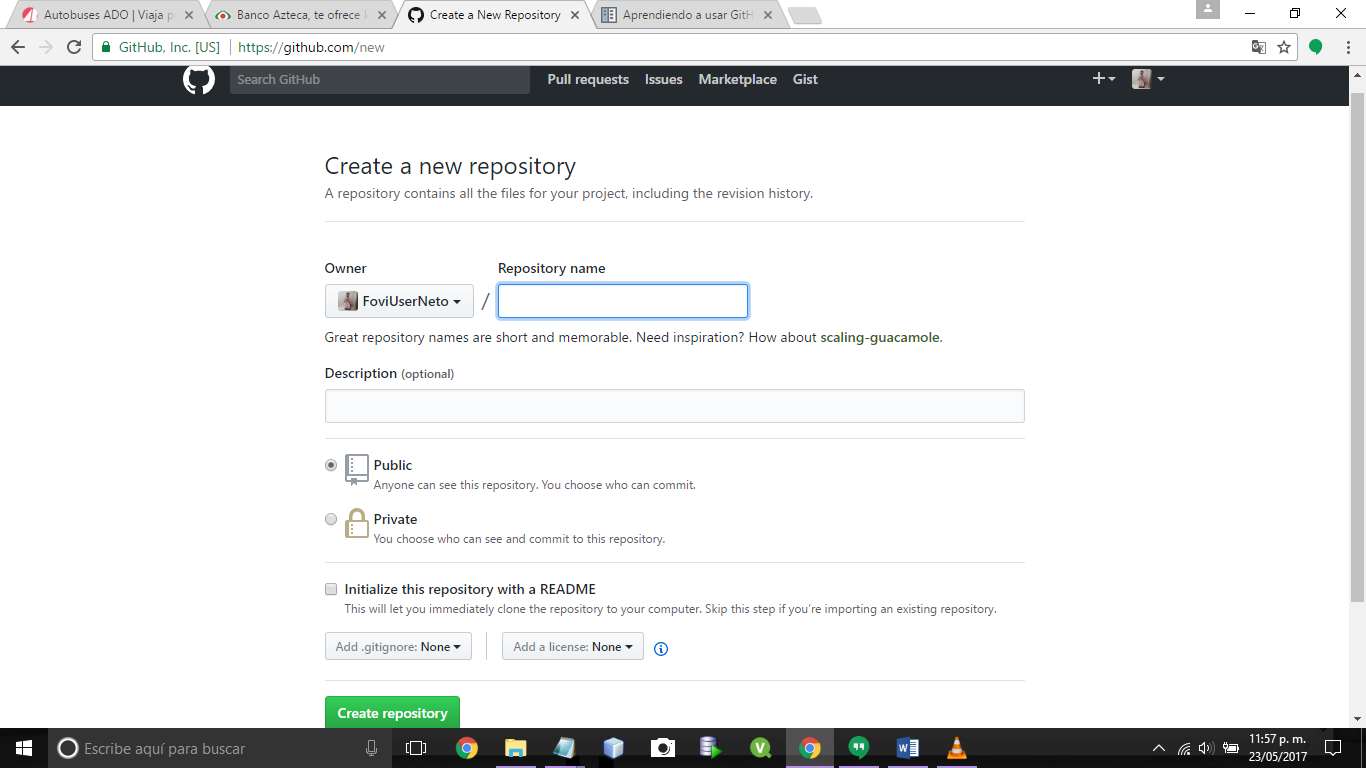


4.- Procedemos a crear un nuevo repositorio, dando un clic en el apartado de New repository



Creamos un nuevo repositorio

5.- Nombramos al repositorio con el nombre que desee, agregamos una descripción de lo que va contener, dejamos en public la casilla, si activamos el private se necesita realizar un pago por el servicio ya sea por mensual o anual, activamos la casilla Initialize this repository with a README y dejamos los valores por default y damos un clic en Create repository.



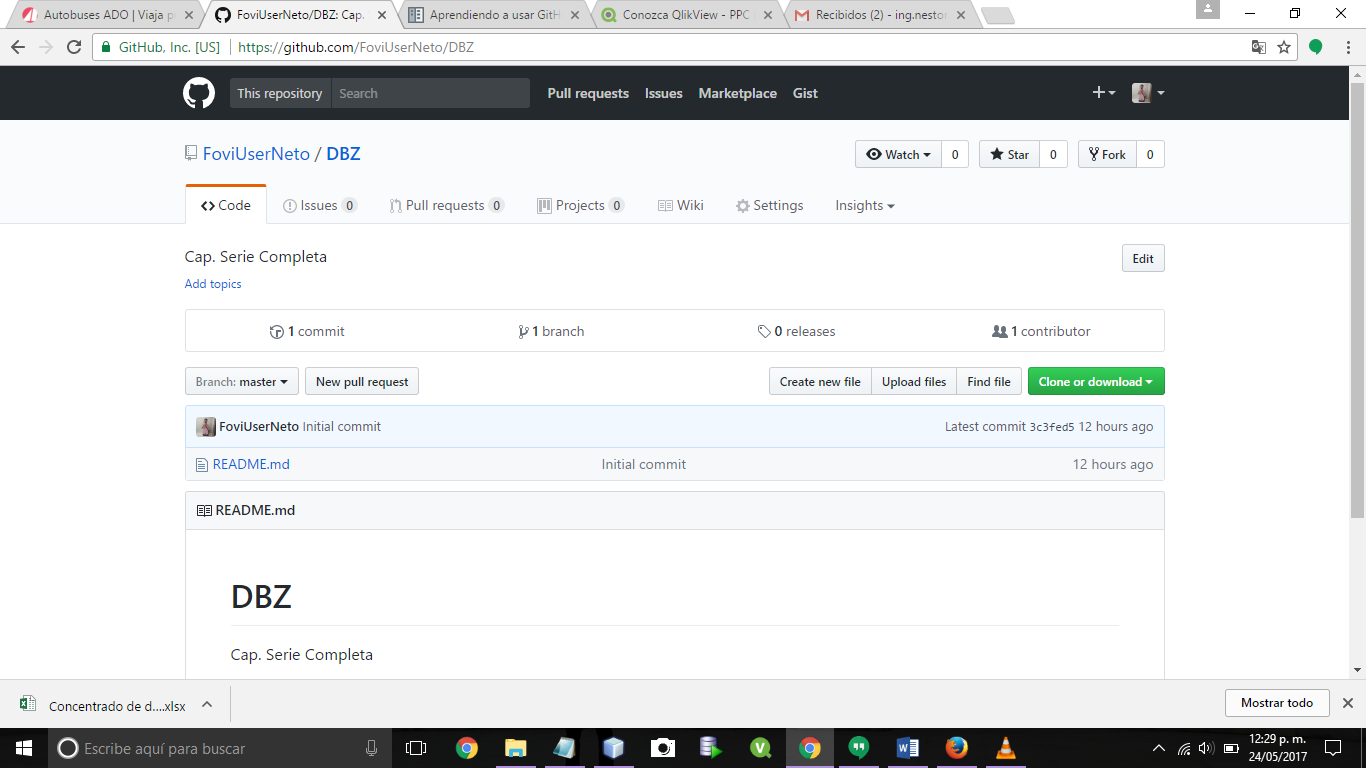
Dejamos los valores por default

Activamos la casilla para generar el archivo README

Dejamos en activado el Public

Agregamos una descripción al repositorio

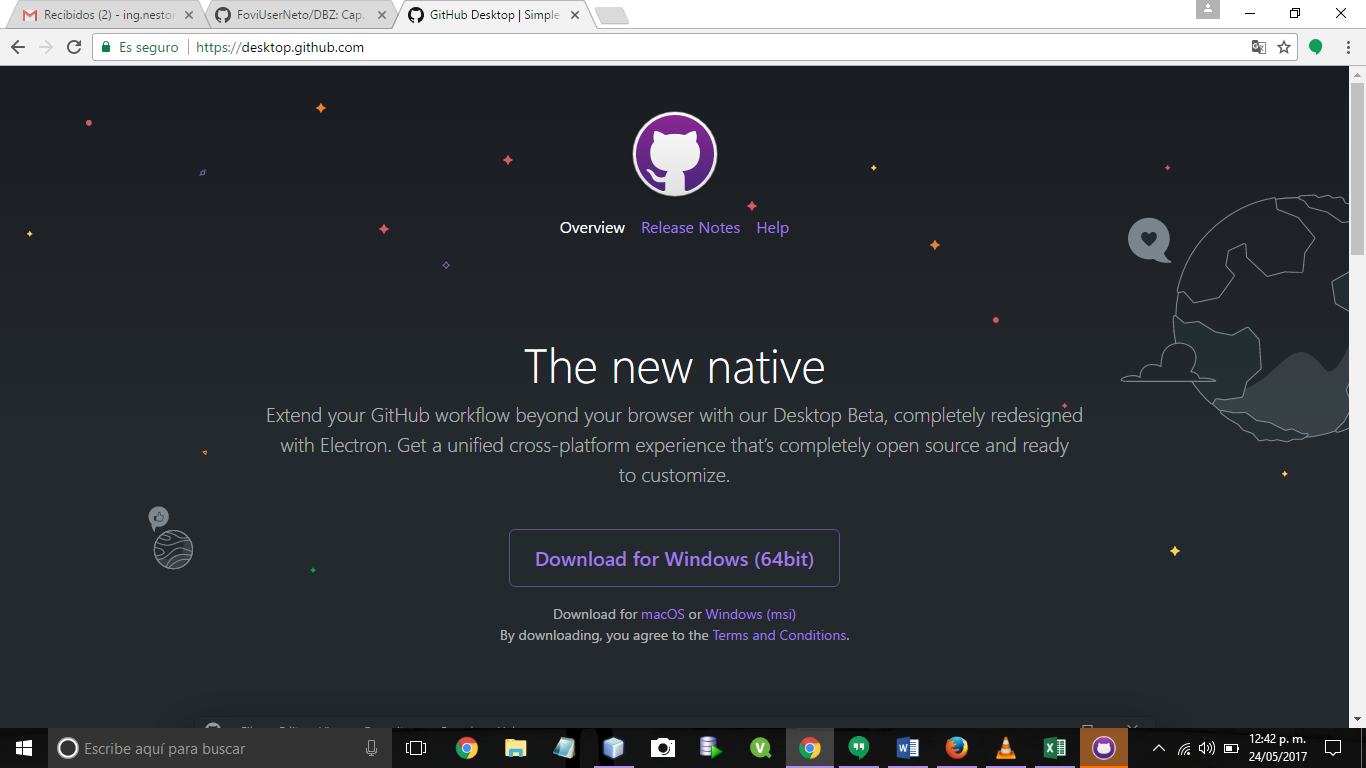
Nombramos el repositorio con el nombre que desee

6.- Al crear el repositorio, inmediatamente nos va a llevar a él. Como nuestro proyecto no tiene nada en su interior, no nos va a mostrar más que una ayuda para subir archivos y proyectos crear, buscar, clonar y descargar las versiones de nuestros archivos

7.- Ahora vamos instalar GitHub para Windows y luego enlazaremos GitHub del ordenador con el GitHub de la Web

Para empezar descargaremos el instalador

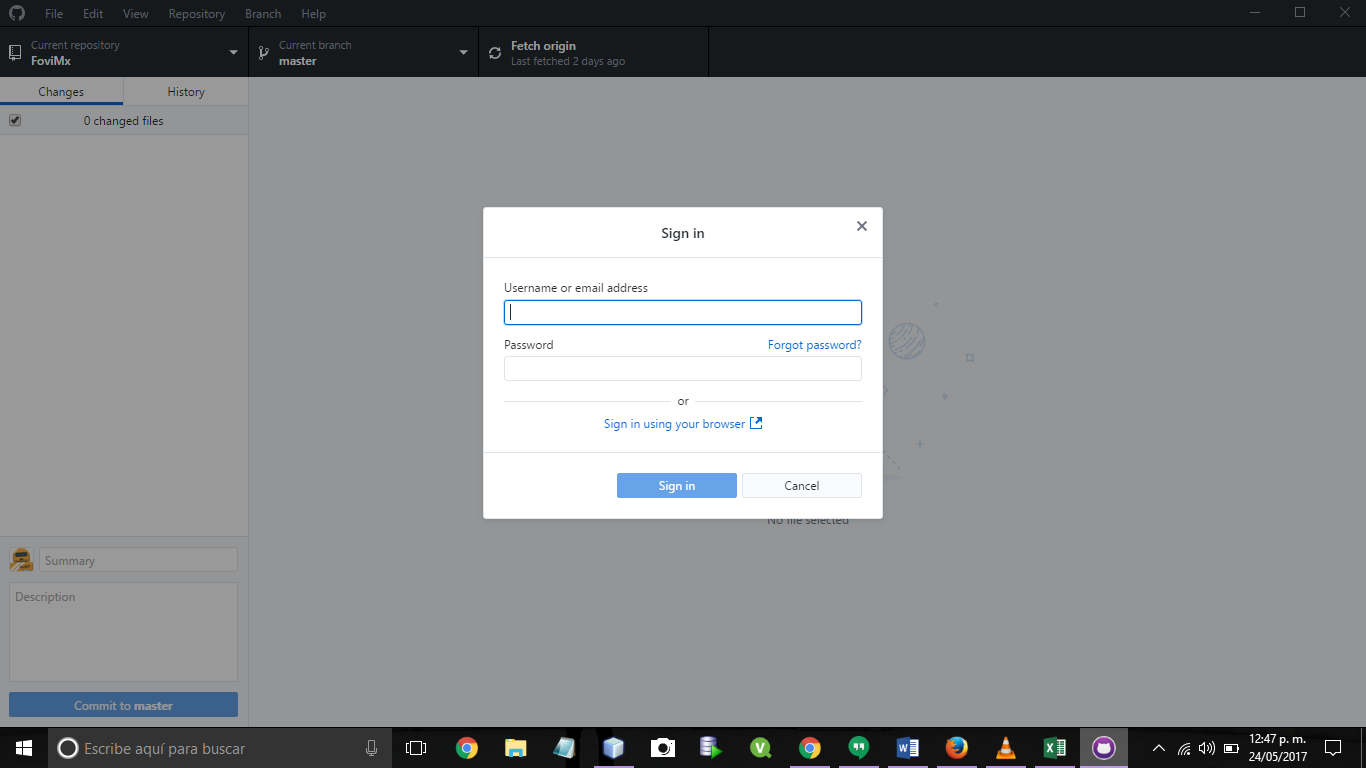
* Ingresamos al link <https://desktop.github.com/>



Ingresamos a este link

Descargamos

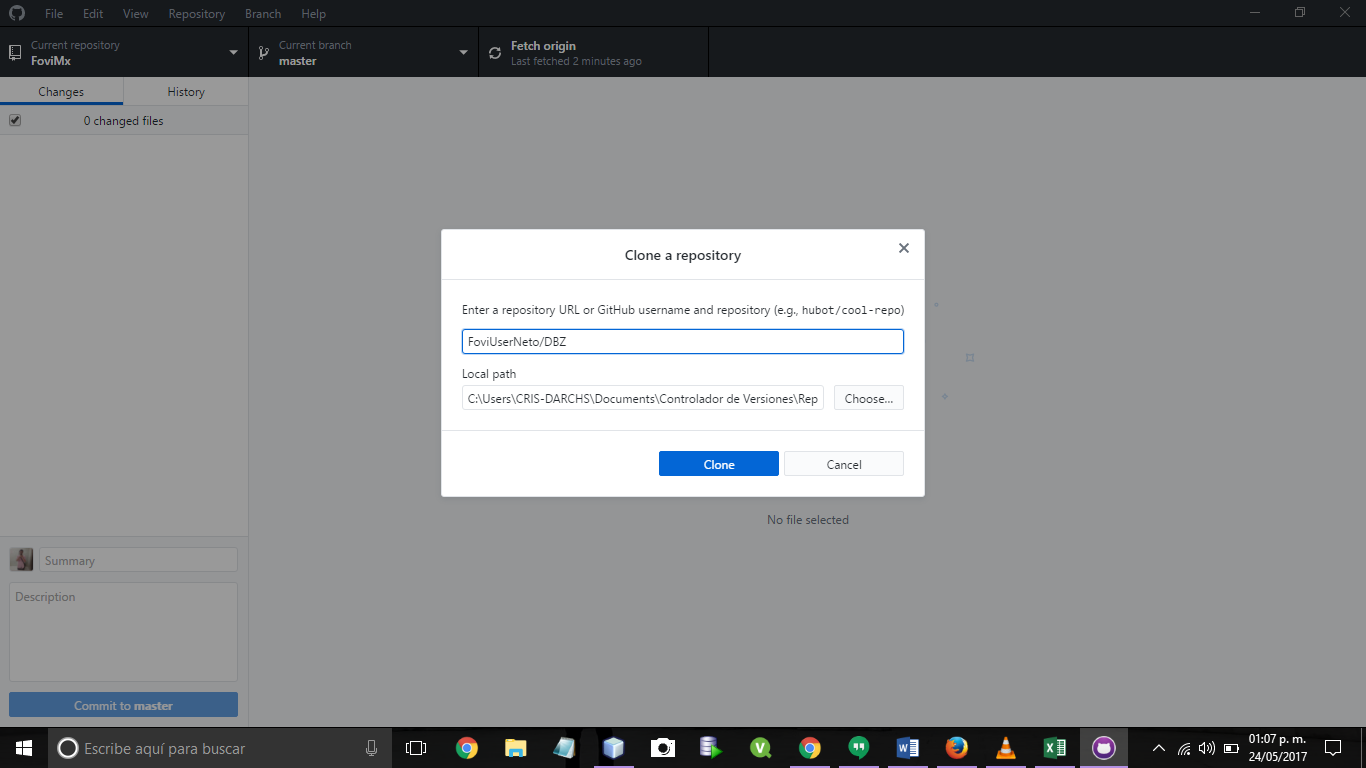
* Una vez instalado el programa, abrimos la aplicación de GitHub, Nos logueamos e ingresamos el usuario o correo y el password y accedemos.



Usuario

Password

* Al ingresar enlazaremos el repositorio creado en GitHub Web con el ordenador para crear una copia exactamente igual de la Web, daremos clic en File, clic en la opción Clone repository, nos saldrá la siguiente ventana en la cual ingresaremos el NombreUsuario/NombreRepositorio y la ruta donde guardaremos los archivos o proyectos.

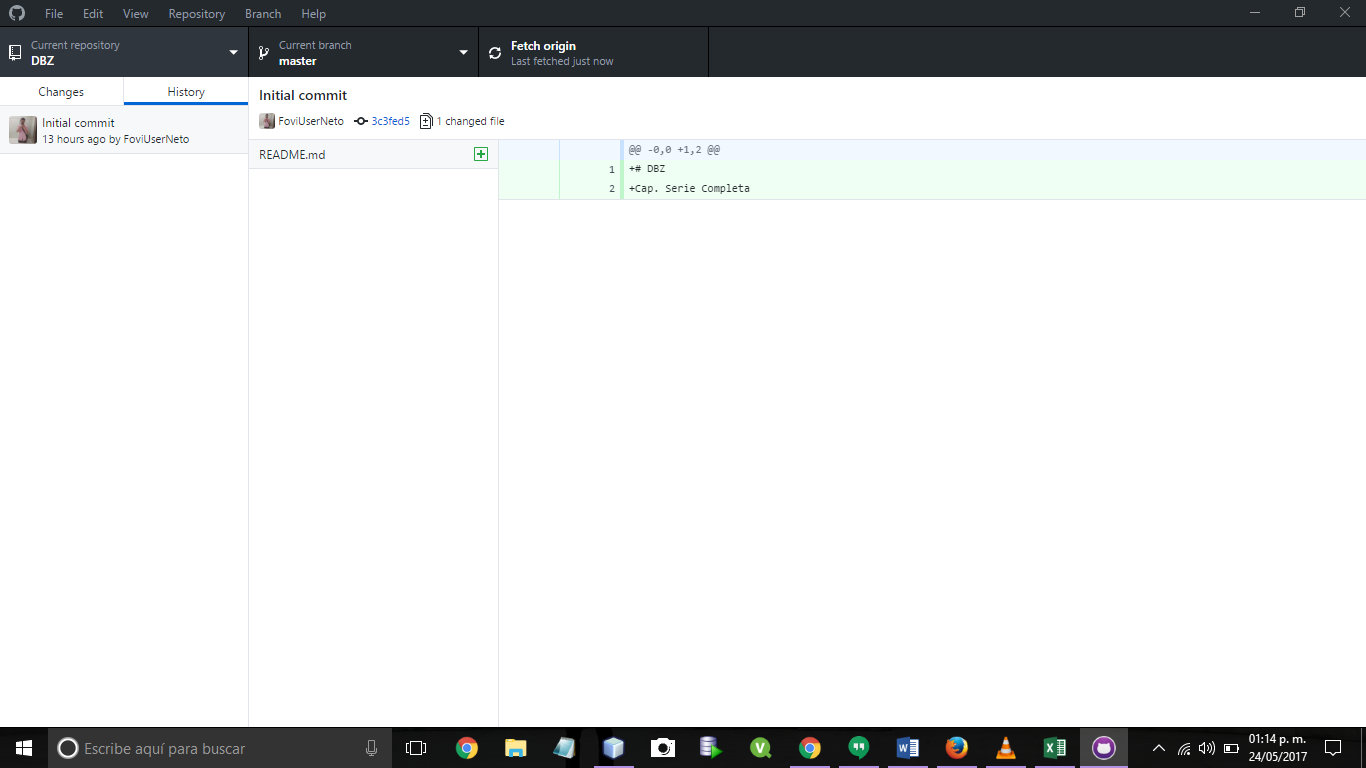


Ruta

NombreUsuario/NombreRepositorio

* Al clonar el repositorio, habremos sincronizado el repositorio del GitHub Web con el ordenador, mostrando el repositorio actual, rama actual, los cambios que se han hecho, el historial, así como los archivos modificados y la visualización del contenido.

Repositorio Actual

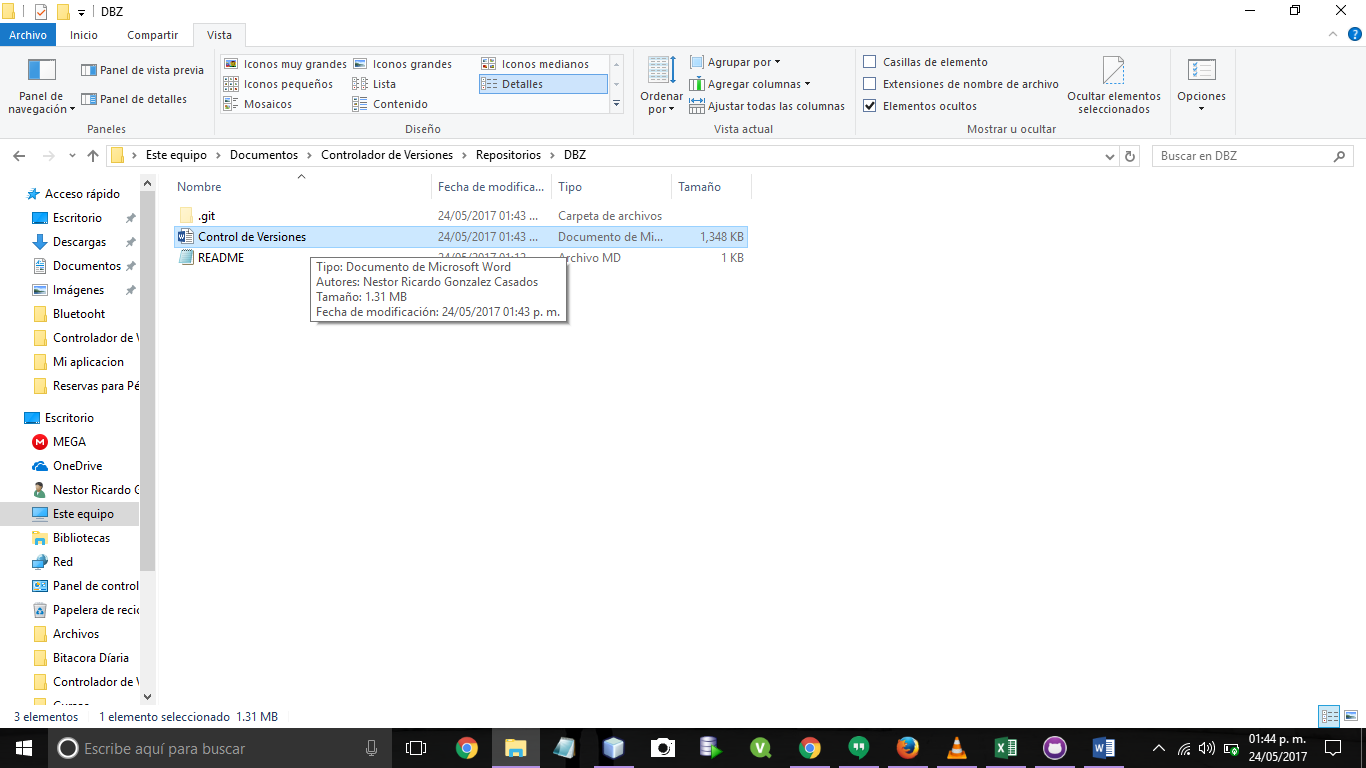
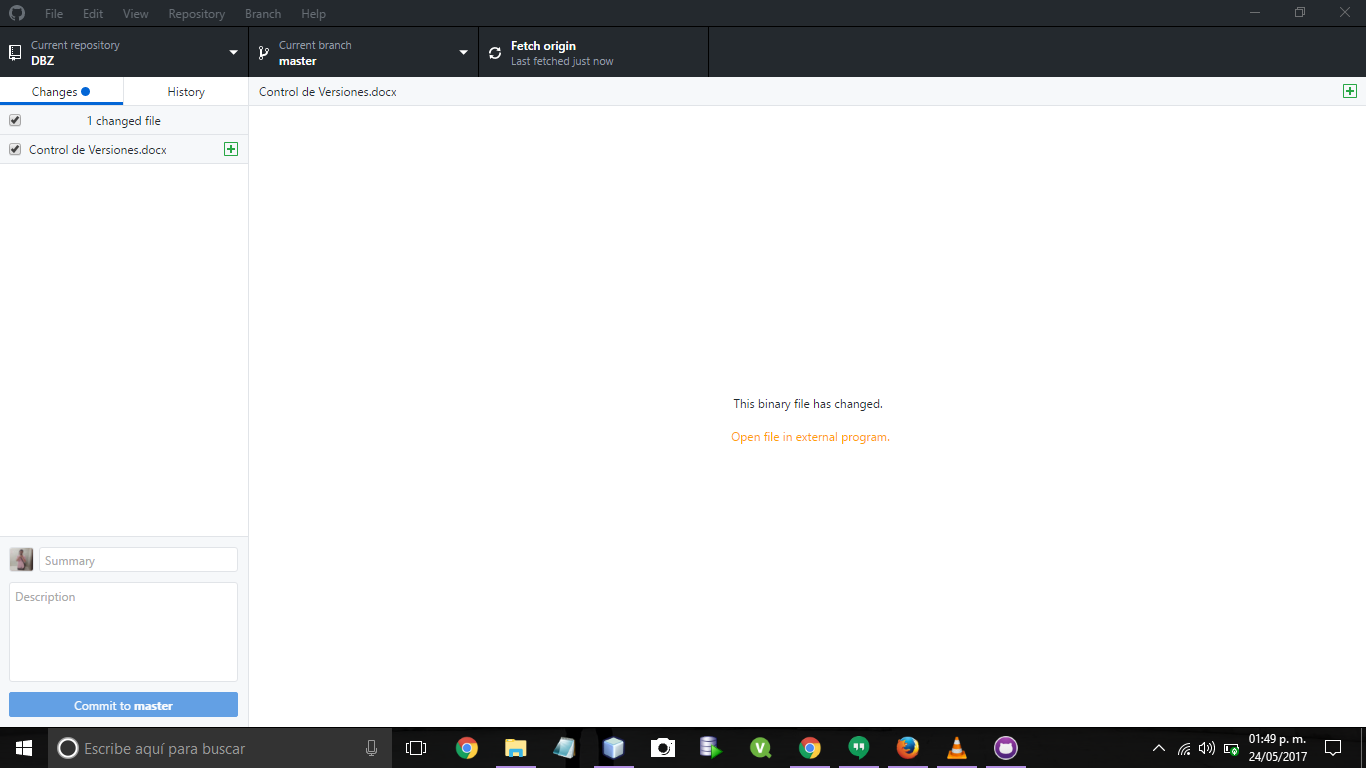


Contenido de archivo

Archivos

Historial

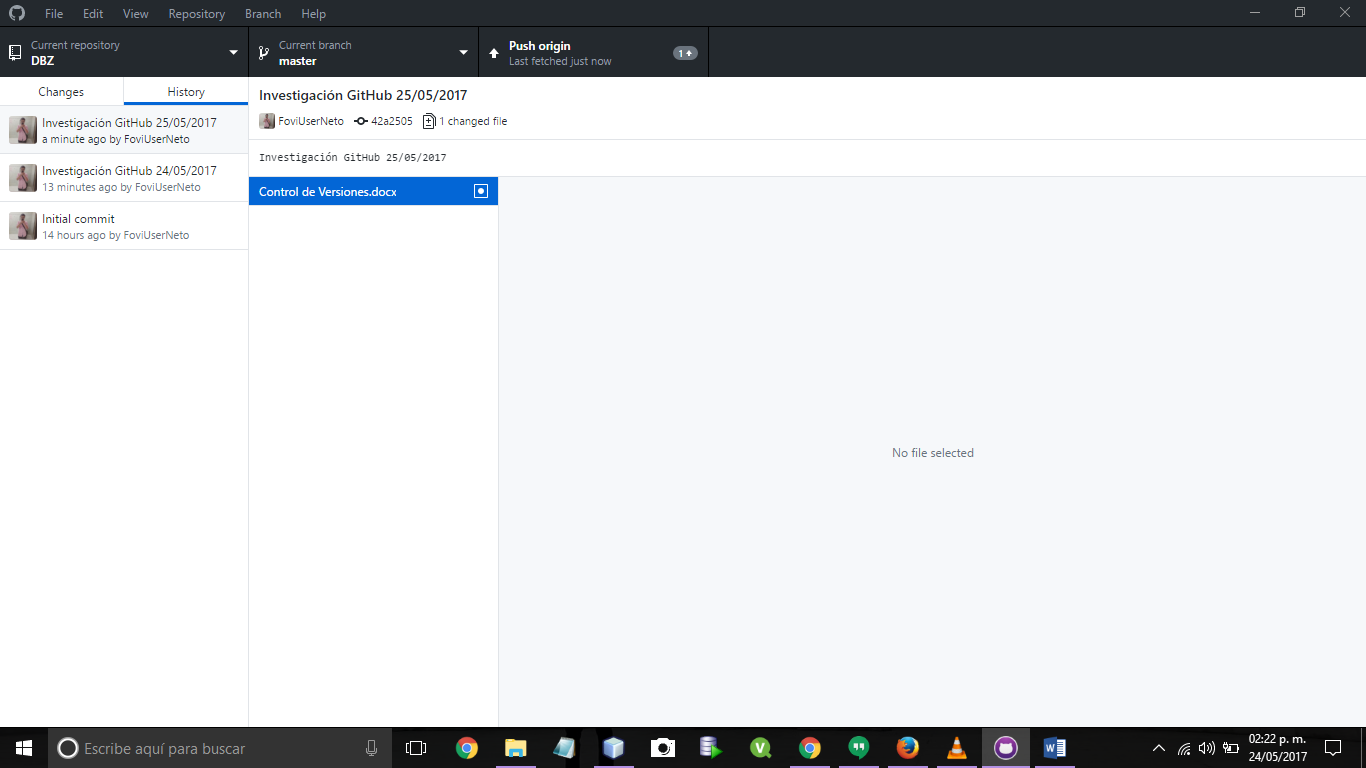
Cambios

* Si vamos al directorio que indicamos, vemos que nos creó una carpeta con el nombre del repositorio actual y genero el archivo README, a partir de aquí podremos agregar los proyectos o documentos que queramos controlar y así poder modificar lo que queramos y guardar cada versión.
* Para nuestro caso agregamos un documento Word y empezamos a trabajar en él, si nos regresamos al GitHub del ordenador, vemos que nos aparece el documento que agregamos en el repositorio, en el cual empezamos a modificar y aparece que tenemos cambios en el documento

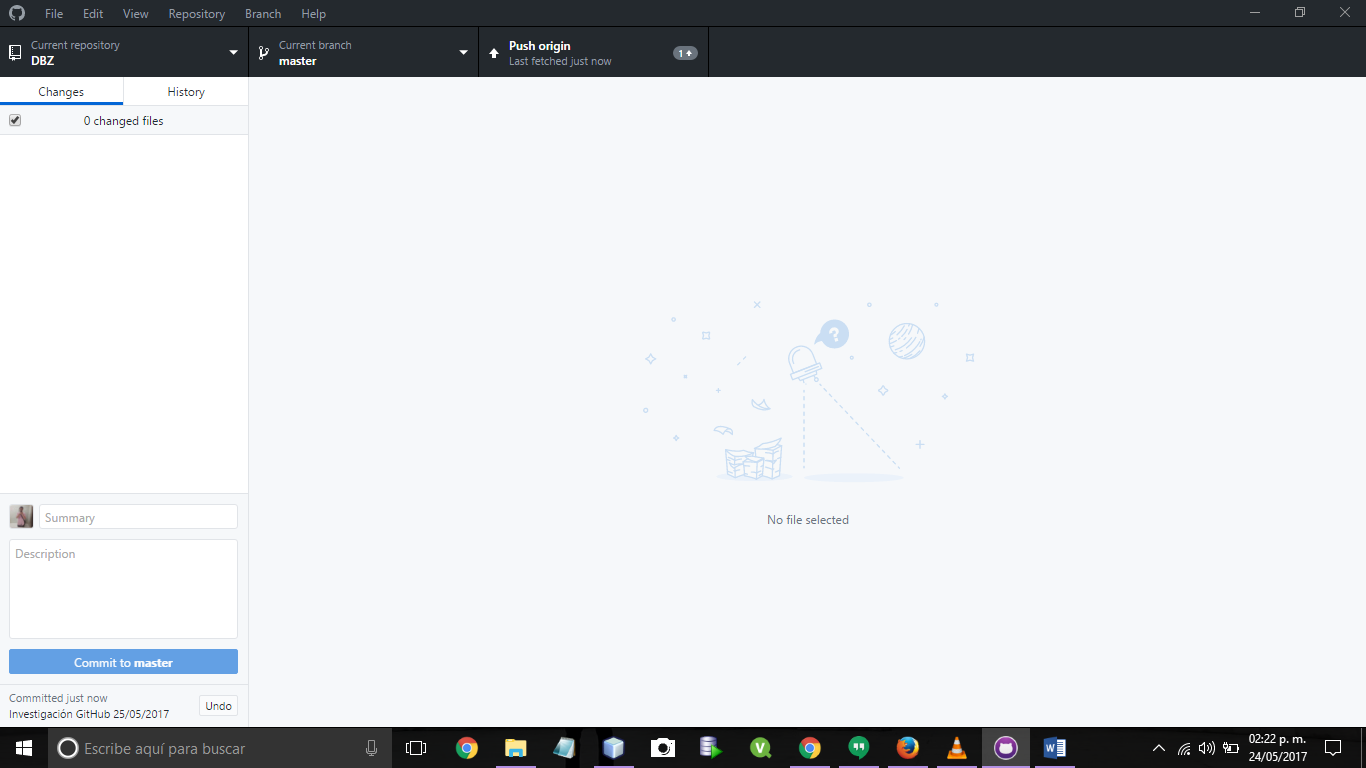
Documento

* Si ya terminamos de modificar el archivo o proyecto, para guardar la versión que tenemos hacemos un Commit to master, pero esta versión se guardara en el ordenador todavía no se guardaran en la nube (Todos los Commits que hagamos se guardaran en el historial), pero antes que nada agregamos un Resumen y una Descripción de la versión que estamos haciendo.
* Al hacer clic en el Commit se guardara la versión y aparecerá una nueva versión en el historial, cabe mencionar que esta versión solamente esta guardada en el ordenador.

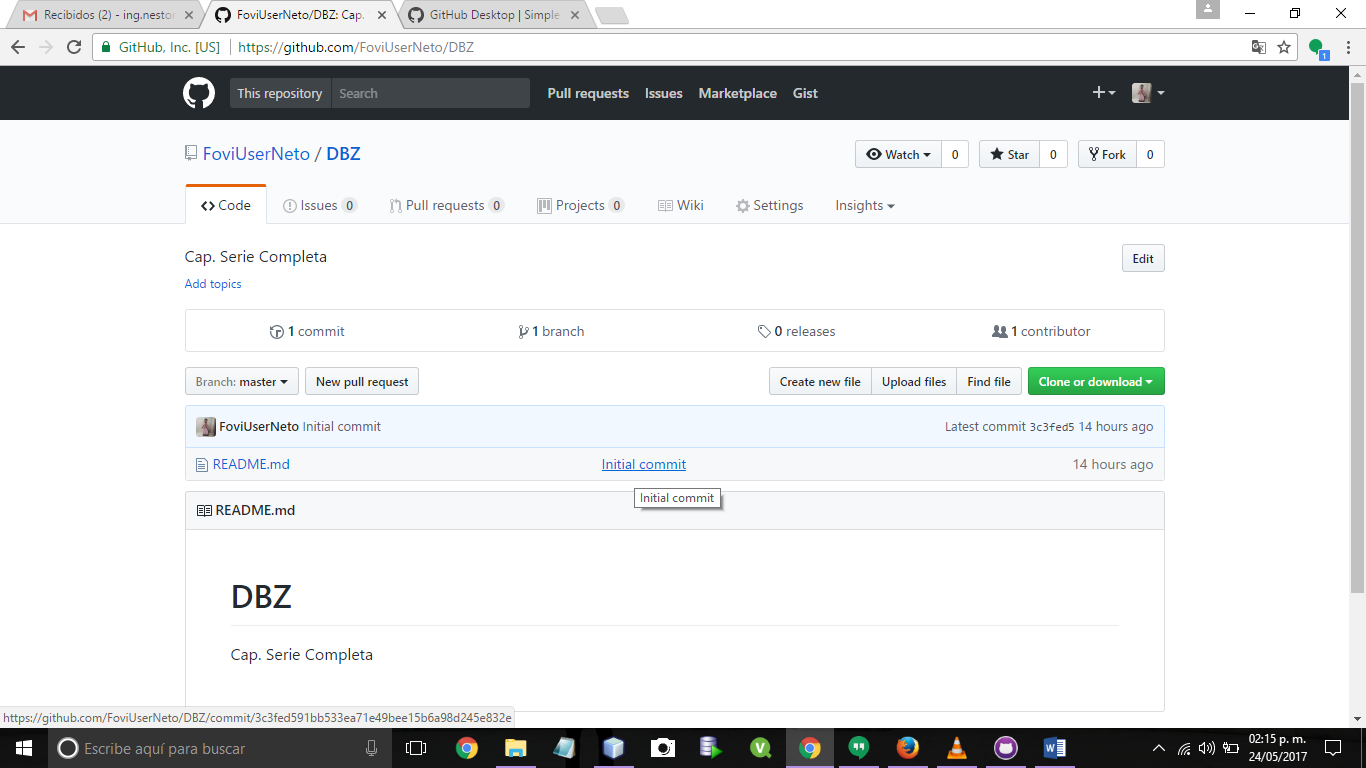
Versión Actual



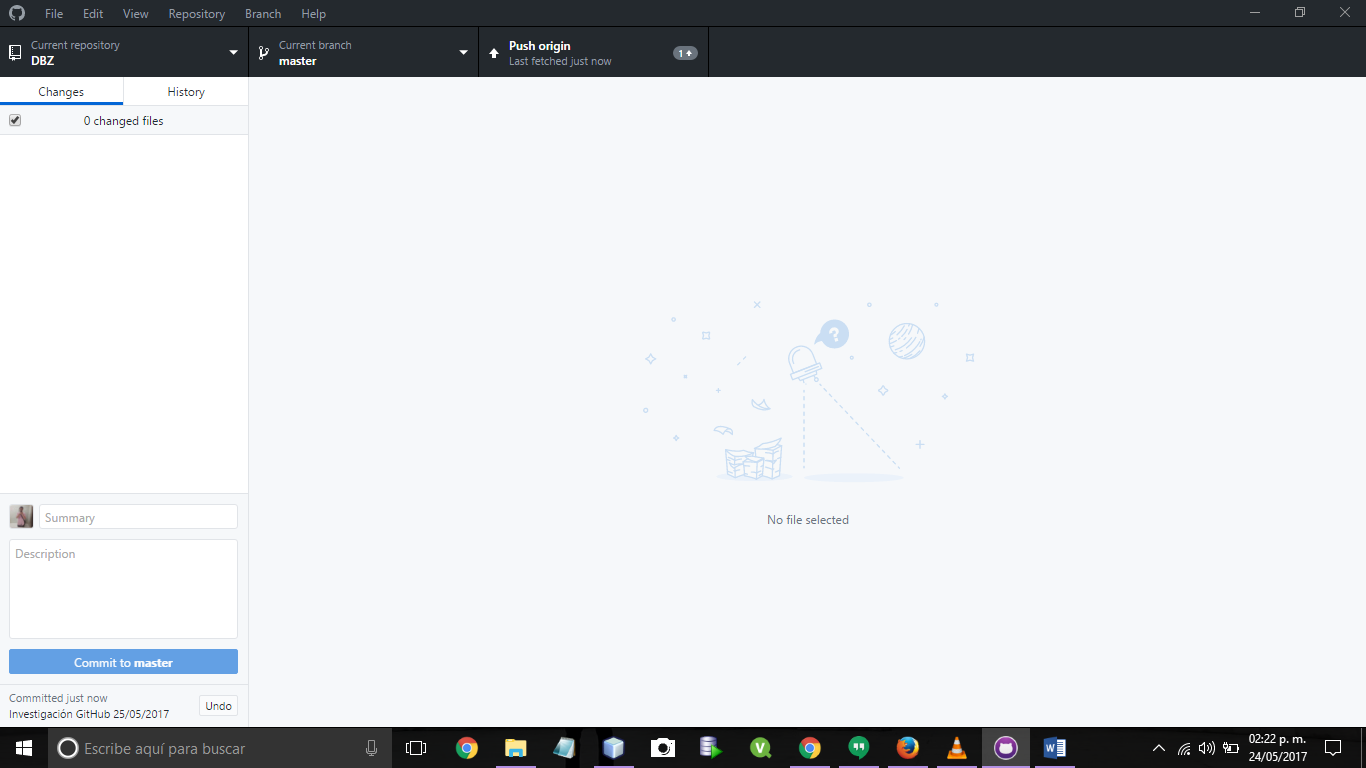
* Vemos que no hay Cambios



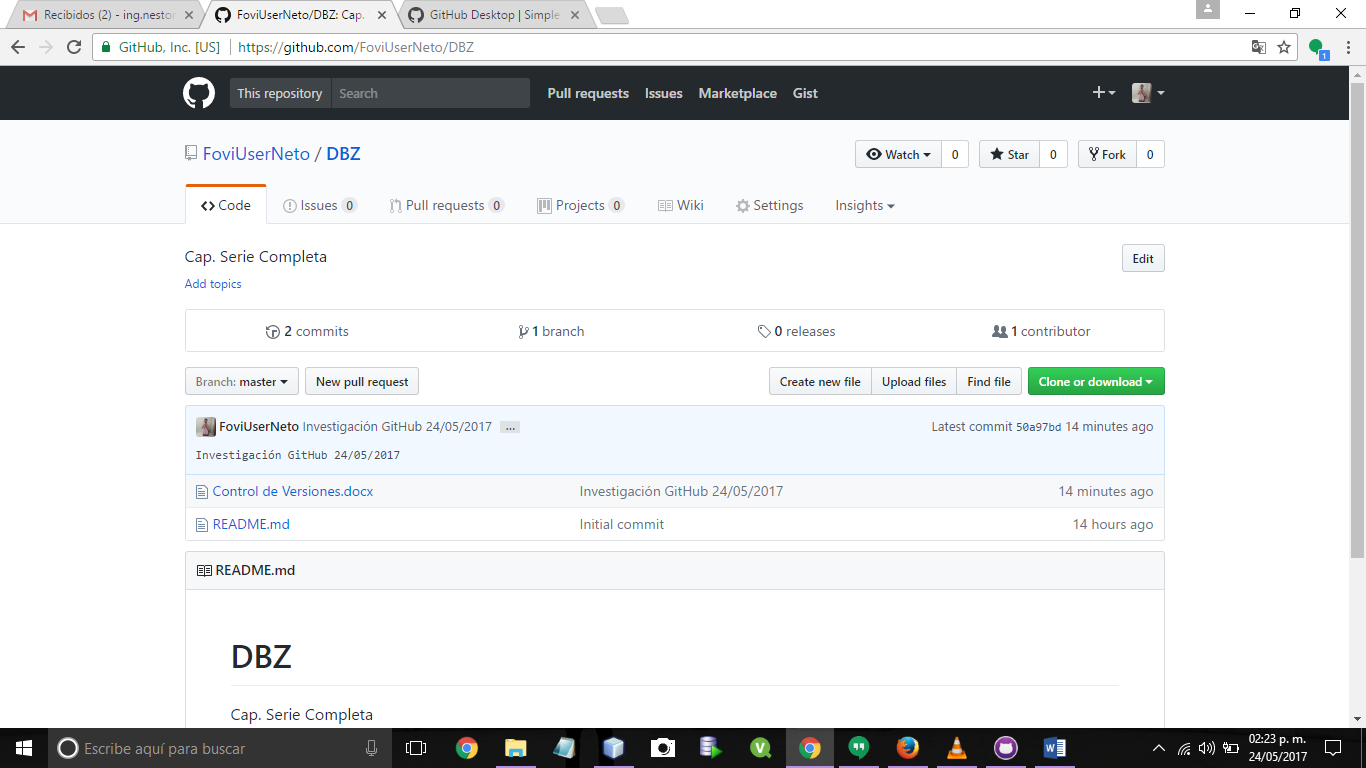
* Si vamos al GitHub Web, vemos que el archivo de Word que estamos trabajando no se ha subido al portal, no se ha subido la versión actual



* Para enviar la versión del documento en la nube hacemos clic en Push origin



* Como vemos al momento de hacer clic en Push origin ya nos aparece el archivo de Word con la versión más reciente, como se puede ver tenemos 2 commits en el repositorio



* Si hacemos clic en los Commits, aparece todas las versiones que hemos realizado