Tutorial sobre GIT & GitHub

Acerca del control de versiones

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones específicas más adelante.

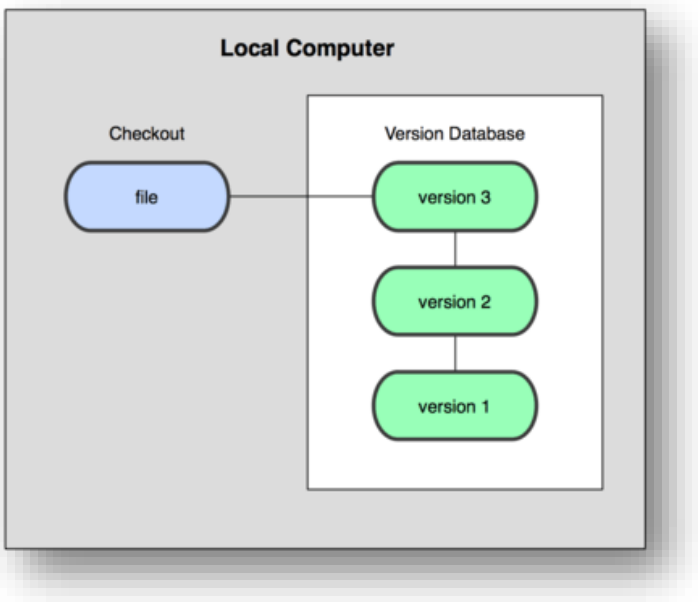
Te permite revertir archivos a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, y mucho más.

Hay diferentes sistemas de control de versiones que pueden clasificarse en: locales, centralizado y distribuido.

Locales

Un método de control de versiones usado por mucha gente es copiar los archivos a otro directorio (quizás indicando la fecha y hora en que lo hicieron). Este enfoque es muy común porque es muy simple, pero también propenso a errores.

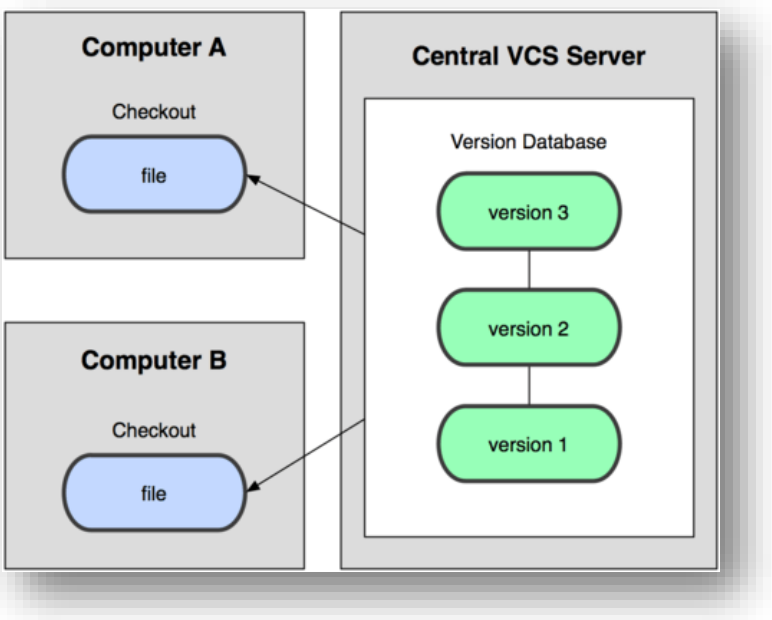
Para hacer frente a este problema, existen los VCSs locales que contienen una simple base de datos en la que se lleva registro de todos los cambios realizados sobre los archivos.



Centralizados

El siguiente gran problema con que se encuentran los desarrolladores es la necesidad de colaborar con otros en un mismo proyecto. Para solventar esto, se desarrollaron los sistemas

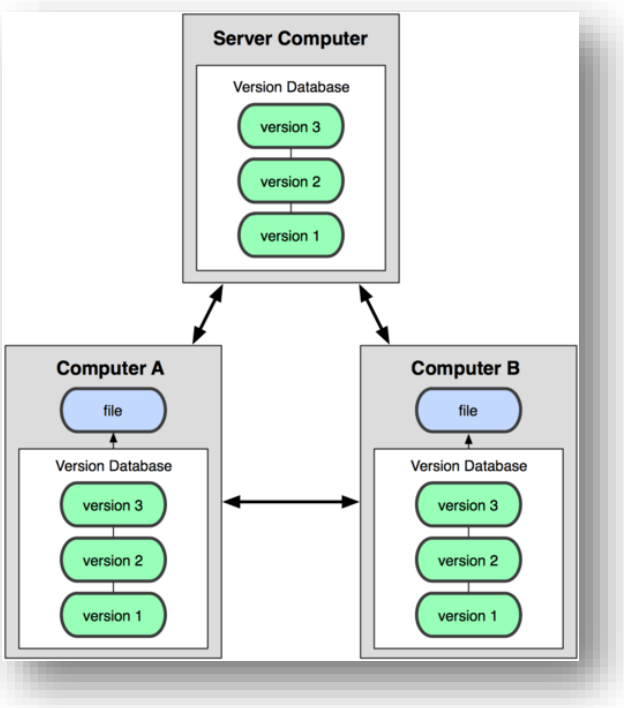
de control de versiones centralizados (Centralized Version Control Systems o CVCSs en inglés). Estos sistemas, como CVS, Subversion, y Perforce, tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos desde ese lugar central. Durante muchos años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



Esta configuración también tiene serias desventajas. La más obvia es el punto único de fallo que representa el servidor centralizado. Si ese servidor se cae durante una hora, entonces durante esa hora nadie puede colaborar o guardar cambios versionados de aquello en que están trabajando. Si el disco duro en el que se encuentra la base de datos central se corrompe, y no se han llevado copias de seguridad adecuadamente, se pierde absolutamente todo —toda la historia del proyecto salvo aquellas instantáneas que la gente pueda tener en sus máquinas locales.

Distribuidos

Los sistemas de control de versiones distribuidos (Distributed Version Control Systems o DVCSs en inglés) resuelven este problema. 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 falla, 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.



Git forma parte de esta categoría.

Instalando GIT

La instalación está muy bien explicada en

http://git-scm.com/book/es/Empezando-Instalando-Git

En Windows se descarga el instalador desde la página de GitHub y se ejecuta (http://msysgit.github.com/).

Se instalará la versión de línea de comandos (incluido un cliente SSH) y la interfaz gráfica de usuario estándar.

Instalando GitHub para windows

Desde la cátedra recomendamos que se instalen la aplicación de GitHub para windows:

- https://windows.github.com/

Instalación y configuración

1. **Configuración del usuario GIT**

Cada miembro del equipo debe establecer un nombre de usuario y dirección de correo electrónico con el cual se identificarán los cambios realizados en el repositorio en común.

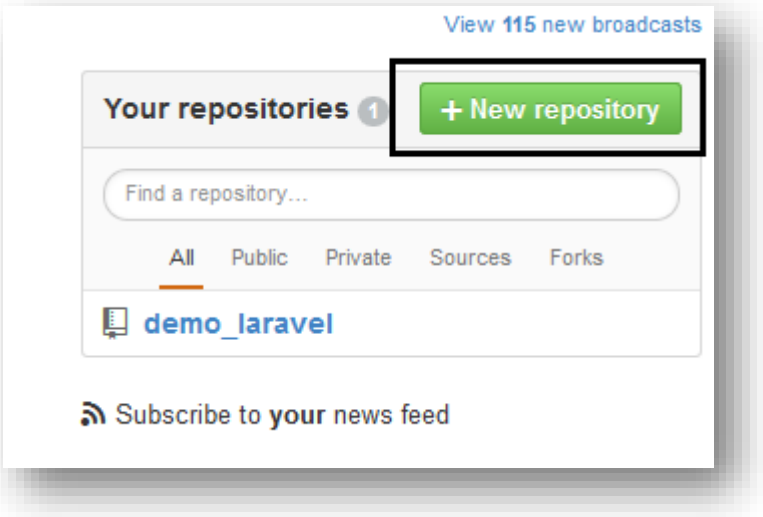
```
$ git config --global user.name "Virginia Ainchil"

$ git config --global user.email virginia@ainchil.com
```

2. Creación de un usuario en GitHub

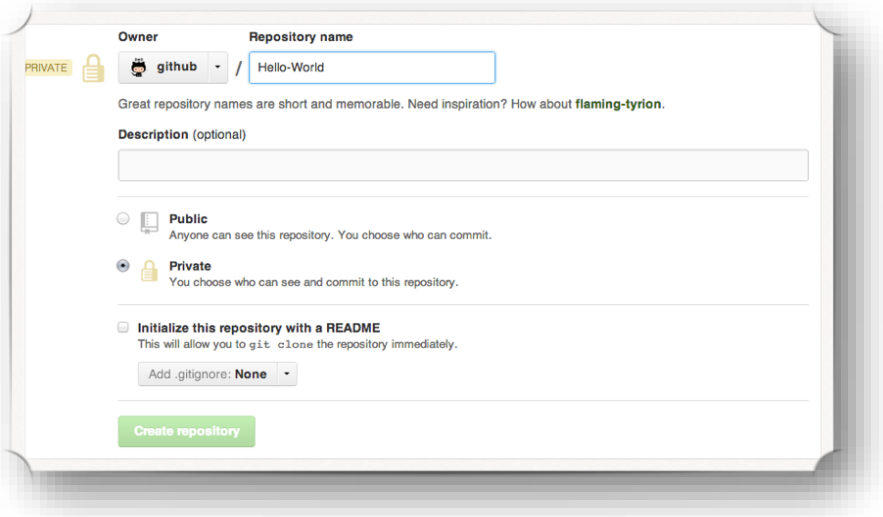
Cada miembro del equipo debe ingresar a github.com y registrar el usuario.

3. Creación de un repositorio

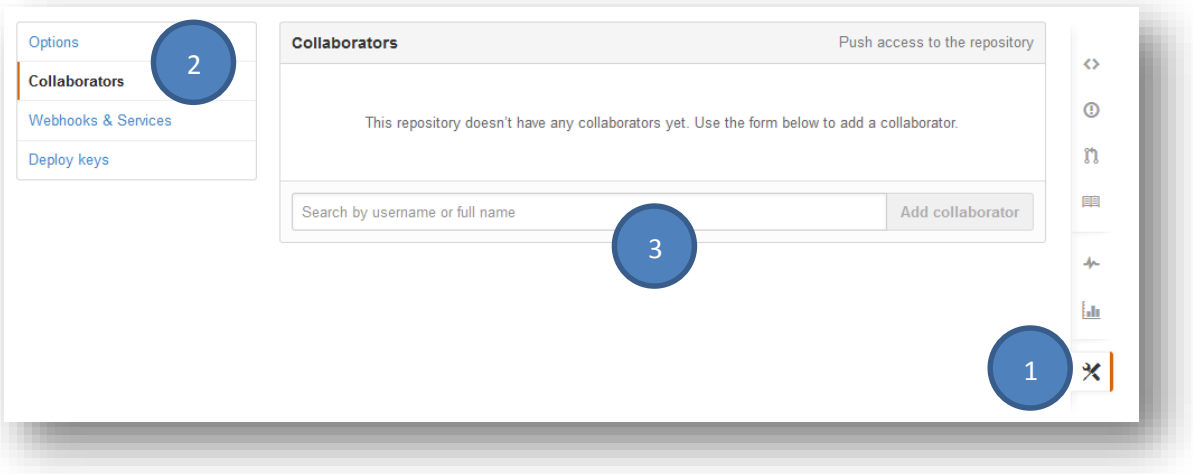
Sólo un integrante del equipo deberá crear el repositorio sobre el que todos trabajarán en GitHub.



Al presionar “New repository”, aparecerá la siguiente pantalla donde se establecerá el nombre del repositorio. Deben elegir la opción de privacidad como *público* y tildar la opción de crear el archivo README. Este archivo deberán usarlo para especificar el número de grupo y los integrantes del mismo:

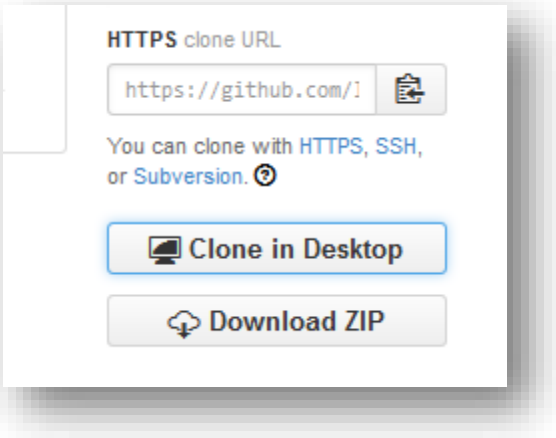


Luego de crear el repositorio, desde las opciones de configuración del mismo (*Settings*), se podrán agregar como colaboradores (con el nombre de usuario que cada uno configuró) al resto de los integrantes del grupo para que puedan realizar cambios sobre el mismo.



4. Clonación del repositorio (Copia local)

Cada integrante del grupo deberá acceder al repositorio (a través de su URL) y hacer clic en "*Clone in Desktop*" para bajarse una copia del mismo.



5. Envío de la URL del repositorio al ayudante

El grupo deberá encargarse de enviar la URL del repositorio a su/s ayudante/s.

Comandos de GIT

Para trabajar con un repositorio Git existen un conjunto de comandos. Los más comunes son:

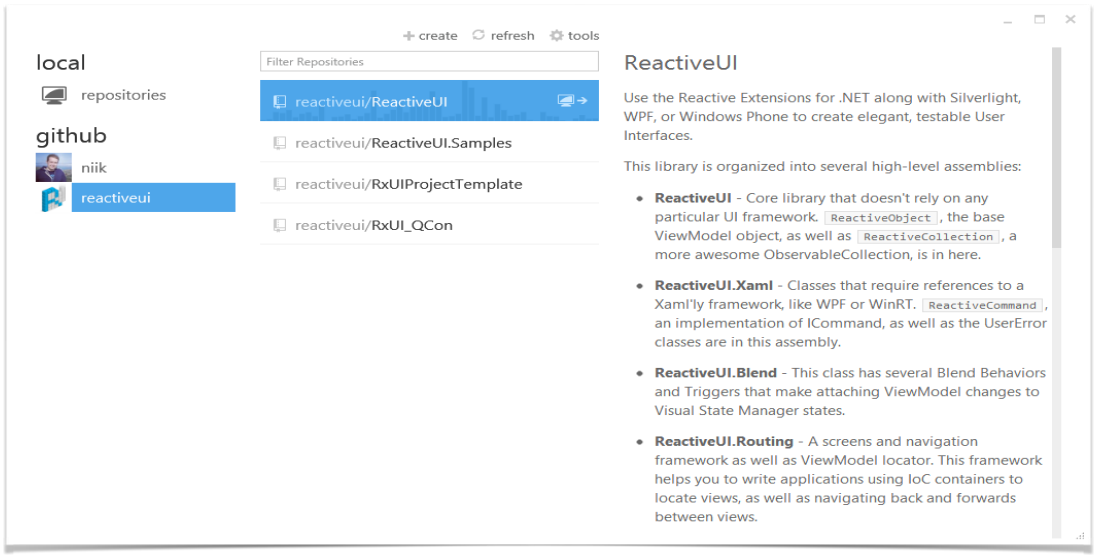
- COMMIT: Confirmar localmente alguna modificación del repositorio.
- PULL: Bajar los cambios del repositorio remoto.

- PUSH: Enviar los cambios al repositorio remoto.

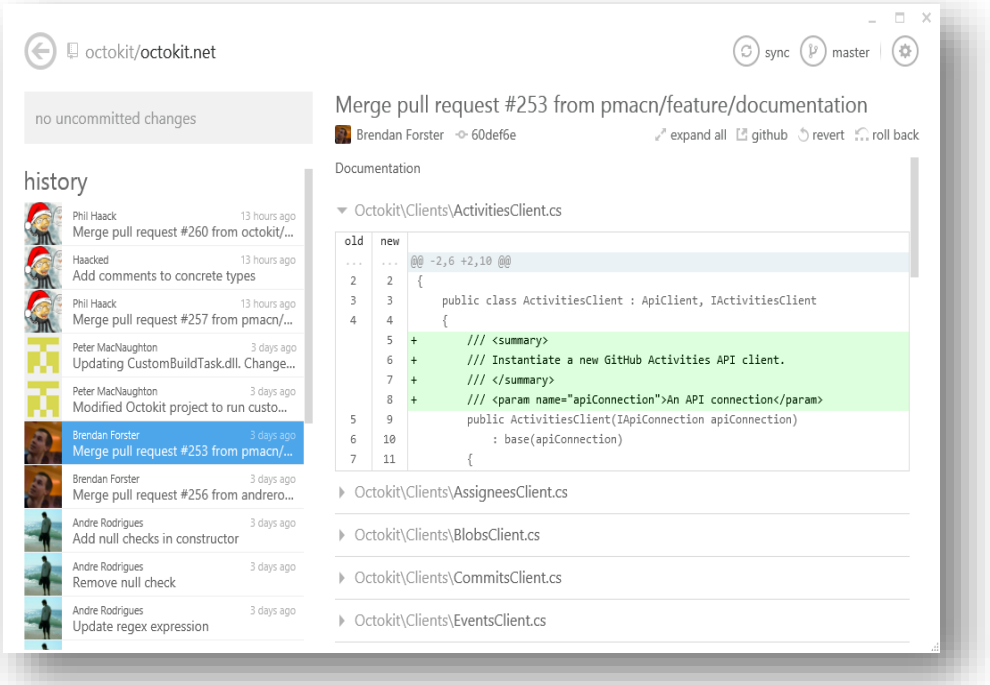
Con la aplicación GitHub para Windows, se puede invocar la función SYNC que hace un PULL y un PUSH.

Utilizando GitHub para Windows

Si se siguieron los pasos explicados previamente, el repositorio clonado aparecerá automáticamente listado en la aplicación.

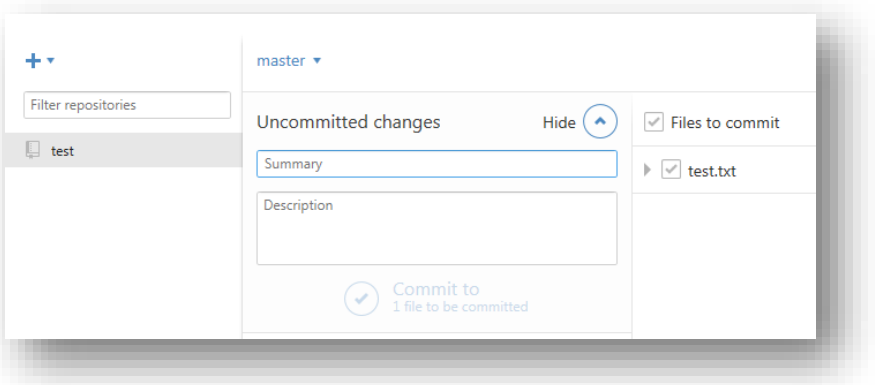


Para ver los cambios que se realizaron sobre el repositorio, se debe hacer clic en el nombre del repositorio.



En esta misma pantalla se reflejarán los cambios propios y del resto de los integrantes del equipo.

La aplicación mostrará los archivos que se modificaron localmente dando la posibilidad de que el usuario ejecute el comando *commit*, ingresando un mensaje que indiquen los cambios realizados.



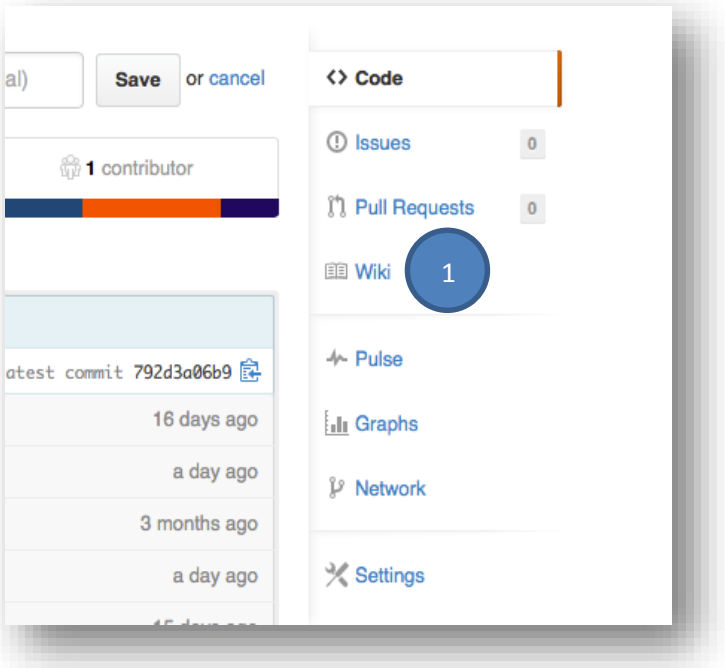
Cada usuario, puede ejecutar el comando *commit* todas las veces que lo crea necesario. Es importante aclarar que estos cambios sólo se reflejan localmente hasta que el usuario ejecute el comando *sync*. También deben usar el comando sync para descargarse los cambios del restos de los integrantes del equipo.

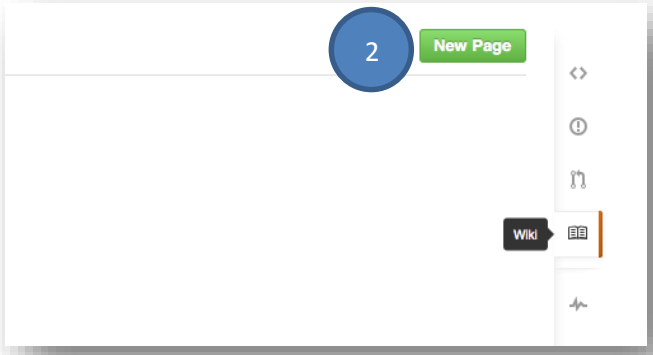
WIKI

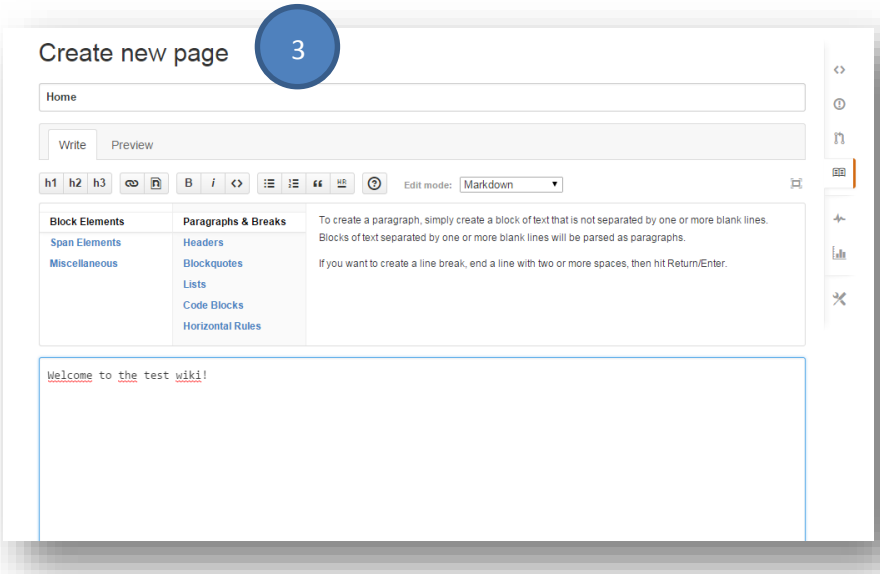
Cada grupo deberá crear una Wiki del proyecto en Github. Como mínimo deberán mantenerse las siguientes páginas:

- Una página descriptiva del proyecto que incluya la descripción del mismo, miembros (nombres y roles) y cualquier otra información relevante. Es importante que en todo momento aparezca claramente el rol de cada miembro, debiéndose actualizar el rol de scrum master en la página cada vez que cambie.
- Una página del proyecto en la que se recoja toda la documentación que se vaya generando (bocetos, decisiones de diseño que vayan tomando, etc)
- Una página en la que se vayan recogiendo las minutas de las reuniones presenciales o virtuales que mantengan.

Creación de la WIKI







Referencias

Git Book (en español) http://git-scm.com/book/es/

GitHub http://github.com

http://rogerdudler.github.io/git-guide/index.es.html

http://nvie.com/posts/a-successful-git-branching-model/