EL VERSIONADO DE SOFTWARE

El versionado de software es el proceso de asignación de un nombre, código o número único, a un software para indicar su nivel de desarrollo.

Generalmente se asigna dos números, *mayor*.*menor* (en inglés: *major*.*minor*), que van incrementando conforme el desarrollo del software aumente y se requiera la asignación de un nuevo nombre, código o número único. Aunque menos habituales, también puede indicarse otro número más, *micro*, y la fase de desarrollo en que se encuentra el software.

Se aumenta el número cuando:

- *mayor*: el software sufre grandes cambios y mejoras.
- *menor*: el software sufre pequeños cambios y/o correcciones de errores.
- *micro*: se aplica una corrección al software, y a su vez sufre pocos cambios.
- *fase*: se indica si se encuentra en una fase de desarrollo que no sea la *final* o *estable*, es decir, una *fase* inestable o en pruebas. Se suele indicar con un guion seguido de la *fase* correspondiente en minúsculas, o un espacio seguido de la *fase*. Puede haber varias *versiones* de una misma *fase*, para indicar el avance en el desarrollo del software pero manteniendo la *fase* para indicar que todavía es inestable, indicándose añadiendo un número al final del nombre de la *fase* que va incrementando conforme se publiquen nuevas *versiones* de esta *fase*.

Existen sistemas que no siguen el versionado tradicional.

Algunos proyectos, como Ubuntu, usan los dígitos para indicar la fecha de lanzamiento.

Otros proyectos, como en arcades, usan códigos varios, finalizando en una fecha y la versión del parche.

En algunos proyectos de desarrollo ágil, como Google Chrome, se usa un versionado similar al tradicional. Aunque los cambios en el dígito mayor indican cambios relevantes en el software, no son tantos como es habitual en el versionado tradicional.

Tipos de repositorio de software

Repositorio. Son sistemas de información que preservan y organizan materiales científicos y académicos como apoyo a la investigación y el aprendizaje; y garantizan el acceso a la información.

Definición

Se considera un sistema de gestión de contenidos, que administra la producción científica en formato digital. Utilizan estándares abiertos para garantizar que sus contenidos seann accesibles y puedan ser buscados y recuperados para su uso

posterior. Son un medio de publicación científica. Ofrece otros servicios complementarios.

Un repositorio contiene permite importar, identificar, almacenar, preservar, recuperar y exportar un conjunto de objetos digitales, desde un portal web.

Objetivo

Recopilar y organizar los documentos digitales de carácter científico, docente e institucional producidos por determinada entidad para el apoyo a la investigación, docencia y aprendizaje. De esta manera se mejora la visibilidad de la producción científica y académica de una universidad o centro de investigación, permitiendo el acceso abierto a sus contenidos y garantizando la preservación y conservación de dicha producción.

Colección de un repositorio

La colección abarca artículos, trabajos científicos, tesis doctorales y de maestría, revistas temáticas, material docente, y otros documentos, en distintos formatos digitales.

Tipos de repositorios

- Repositorio de software: la variedad del servicio que ofertan depende del tipo de licencia usada:
  - Licencia privativa: el administrador limita o restringe las propiedades del software. Ejemplo: Windows Update.
  - Licencia de uso libre: ofrecen una plataforma de trabajo colaborativo y compartida de conocimiento libre sobre cualquier temática, sin ningún tipo de restricciones. Ejemplo: repositorios de software libre, paquetes para el sistema operativo GNU/Linux, desde plataformas como SourceForge o Forja de Guadalinex.
- Repositorios institucionales: desarrollado por organismos políticos, sociales y educativos como universidades e institutos o asociaciones, para depositar, usar y preservar la producción científica y académica que generan en formato digital y haciéndola accesible al público. De esta manera la institución ofrece un servicio acorde al movimiento de acceso abierto.
- Repositorios temáticos: creados por un grupo de investigadores, una institución, etc. que reúnen documentos relacionados con un área temática particular. La temática suele ser social, de educación ciudadana o académica.
- Repositorios de datos: repositorios que almacenan, conservan y comparten los datos de las investigaciones.

Pasos para la definición de los repositorios

El repositorio institucional no es solo una base de datos y un software, sino que comprende un conjunto de servicios para aquellos que almacenan contenidos y los usuarios finales a los que está destinado el servicio. A continuación se enumeran una serie de aspectos a tener en cuenta en el momento de concebir el servicio. [1]Para ello deben definirse:

1. La misión del servicio.
2. Tipo de contenidos que aceptará.
3. Quiénes son los usuarios principales y las partes interesadas.
4. ¿Qué servicios ofrecería si tuviera recursos ilimitados?
5. Qué puede permitirse ofrecer
6. Si cobrará o no por los servicios
7. Qué responsabilidades tendrá la biblioteca con la comunidad de contenidos.
8. Cuáles son sus principales prioridades de servicio.
9. Cuáles son sus prioridades a corto y a largo plazo.

Beneficios

1. Permite el acceso abierto a los resultados de la actividad científica y académica.
2. Captura, identificación, almacenamiento, conservación y recuperación de sus contenidos digitales.
3. Permite a los investigadores identificar los autores que abordan temáticas afines a su especialidad y/o líneas de investigación.
4. Estimula la cooperación científica entre las diferentes disciplinas y comunidades.
5. Reutilización de contenidos.
6. Ofrece la difusión más amplia posible de toda la oferta de la producción intelectual digital generada en una institución.
7. Aumentar la visibilidad de sus investigadores, ampliando la difusión y el uso de sus trabajos.
8. Estimula la innovación y el aprendizaje.
9. Facilitan un análisis cualitativo del trabajo de sus miembros.
10. Aumentan la visibilidad de las investigaciones.
11. Reconocen y dan acceso a los objetos digitales científicos no incluidos en los canales tradicionales de publicación.
12. Mejoran la comunicación científica y hacen avanzar la investigación permitiendo a los usuarios localizar y recuperar información relevante más rápida y fácilmente.

Git, es un software de control de versiones diseñado por Linus Torvalds. La pregunta es ¿qué es control de versiones? Pues bien, se define como control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo es decir a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración, y para los que aún no les queda claro del todo, control de versiones es lo que se hace al momento de estar desarrollando un software o una página web. Exactamente es eso que haces cuando subes y actualizas tu código en la nube, o le añades alguna parte o simplemente le editas cosas que no funcionan como deberían o al menos no como tú esperarías.

Y, entonces ¿a que le llamamos sistema de control de versiones? Muy sencillo, son todas las herramientas que nos permiten hacer todas esas modificaciones antes mencionadas en nuestro código y hacen que sea más fácil la administración de las distintas versiones de cada producto desarrollado; es decir Git.

Git

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente, es decir Git nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida y por trabajo nos referimos a algún software o página que implique código el cual necesitemos hacerlo con un grupo de personas.

Algunas de las características más importantes de Git son:

- Rapidez en la gestión de ramas, debido a que Git nos dice que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.
- Gestión distribuida; Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.
- Gestión eficiente de proyectos grandes.
- Realmacenamiento periódico en paquetes.

GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora. El software que opera GitHub fue escrito en Ruby onRails. Desde enero de 2010, GitHub opera bajo el nombre de *GitHub, Inc.*Anteriormente era conocida como *LogicalAwesome LLC*. El código de los proyectos alojados en GitHub se almacena típicamente de forma pública, aunque utilizando una cuenta de pago, también permite hospedar repositorios privados.

Bitbucket es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de versiones Mercurial y Git. Bitbucket ofrece planes comerciales y gratuitos. Se ofrece cuentas gratuitas con un número ilimitado de

repositorios privados (que puede tener hasta cinco usuarios en el caso de cuentas gratuitas) desde septiembre de 2010,[1] los repositorios privados no se muestran en las páginas de perfil - si un usuario sólo tiene depósitos privados, el sitio web dará el mensaje *"Este usuario no tiene repositorios"*. El servicio está escrito en Python.[2]

Es similar a GitHub, que utiliza Git. En una entrada de blog del 2008,[3] Bruce Eckel hace una comparación favorablemente de Bitbucket frente a Launchpad, que utiliza Bazaar.

Publicar una página web en Github Pages

5 minuto de lectura

Esté artículo lo escribí originalmente en septiembre de 2013. Como el servicio GitHub Pages ha sufrido algunos cambios en su configuración, vuelvo a publicarlo con las modificaciones oportunas.

Github Pages es un servicio que te ofrece Github para publicar de una manera muy sencilla páginas web. Disponemos de la opción de generar de forma automática las páginas utilizando una herramienta gráfica, pero en este artículo nos vamos a centrar en la creación y modificación de páginas web usando la línea de comandos con el comando git.

Tenemos dos alternativas para crear una página web con esta herramienta:

- Páginas de usuario u organización: Es necesario crear un repositorio especial donde se va a almacenar todos el contenido del sitio web. Si por ejemplo el nombre de usuario de Github es `josedom24`, el nombre del repositorio debe ser `josedom24.github.io`. Todos los ficheros que se van a publicar deben estar en la rama "master". Por último indicar que la URL para acceder a la página sería http://josedom24.github.io.
- Páginas de proyecto o repositorio: A diferencia de las anteriores están asociada a cualquier repositorio que tengamos en Github (por ejemplo supongamos que el repositorio se llama `prueba`). ~~En este caso los ficheros que se van a publicar deben estar en una rama del proyecto llamada `gh-pages`~~

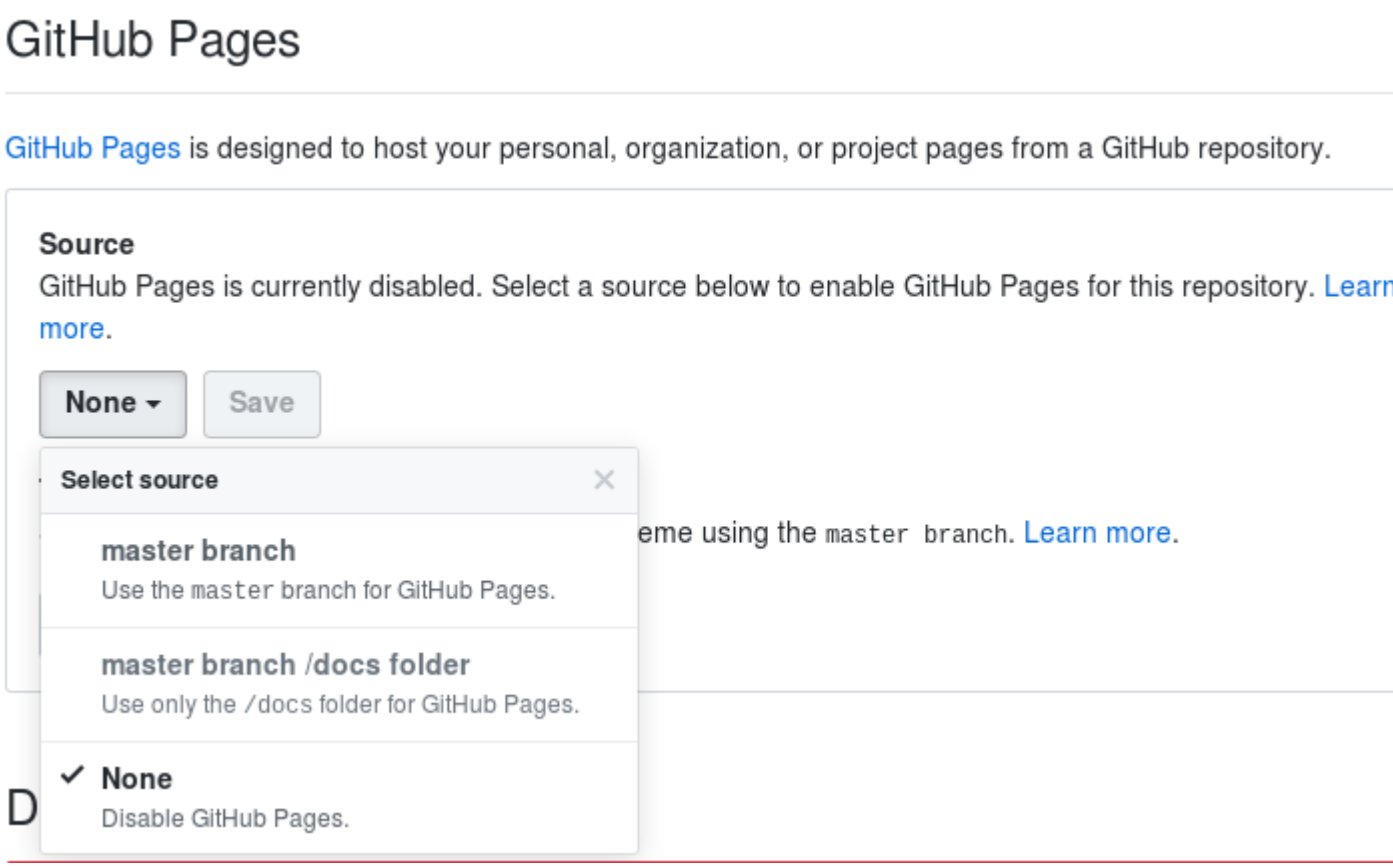
(Actualización 20/9/2017: Actualmente se pueden publicar páginas web en GitHub Pages desde la rama `master`, `gh-pages` o la carpeta `/cod` de la rama `master`).

La URL de acceso al sitio será `http://josedom24.github.io/prueba`.

Creación manual de páginas

Mientras que las páginas de usuario son fáciles de crear, ya que simplemente debemos crear el repositorio y clonarlo en nuestro equipo (git clone) y empezar a crear ficheros que estarán guardados en la rama master, las páginas de repositorio pueden ser un poco más complejas ~~ya que hay que crear la nueva rama que tenemos que llamar gh-pages~~, siguiendo el manual de Github Pages los pasos a dar son los siguientes:

Actualización 20/9/2017: En la configuración del repositorio, podemos escoger donde vamos a guardar nuestra página web: la rama `master`, `gh-pages`(si el repositorio tiene dicha rama) o la carpeta `/cod` de la rama `master`.



En el siguiente ejemplo vamos a crear una rama `gh-pages`, aunque como hemos indicado anteriormente nos serviría la rama `master`:

```
$ git clone https://github.com/user/repository.git
# Clone ourrepository
# Cloninginto 'repository'...
# remote: Countingobjects: 2791, done.
# remote: Compressingobjects: 100% (1225/1225), done.
# remote: Total 2791 (delta 1722), reused 2513 (delta 1493)
```

```
# Receivingobjects: 100% (2791/2791), 3.77 MiB | 969 KiB/s, done.
# Resolving deltas: 100% (1722/1722), done.
```

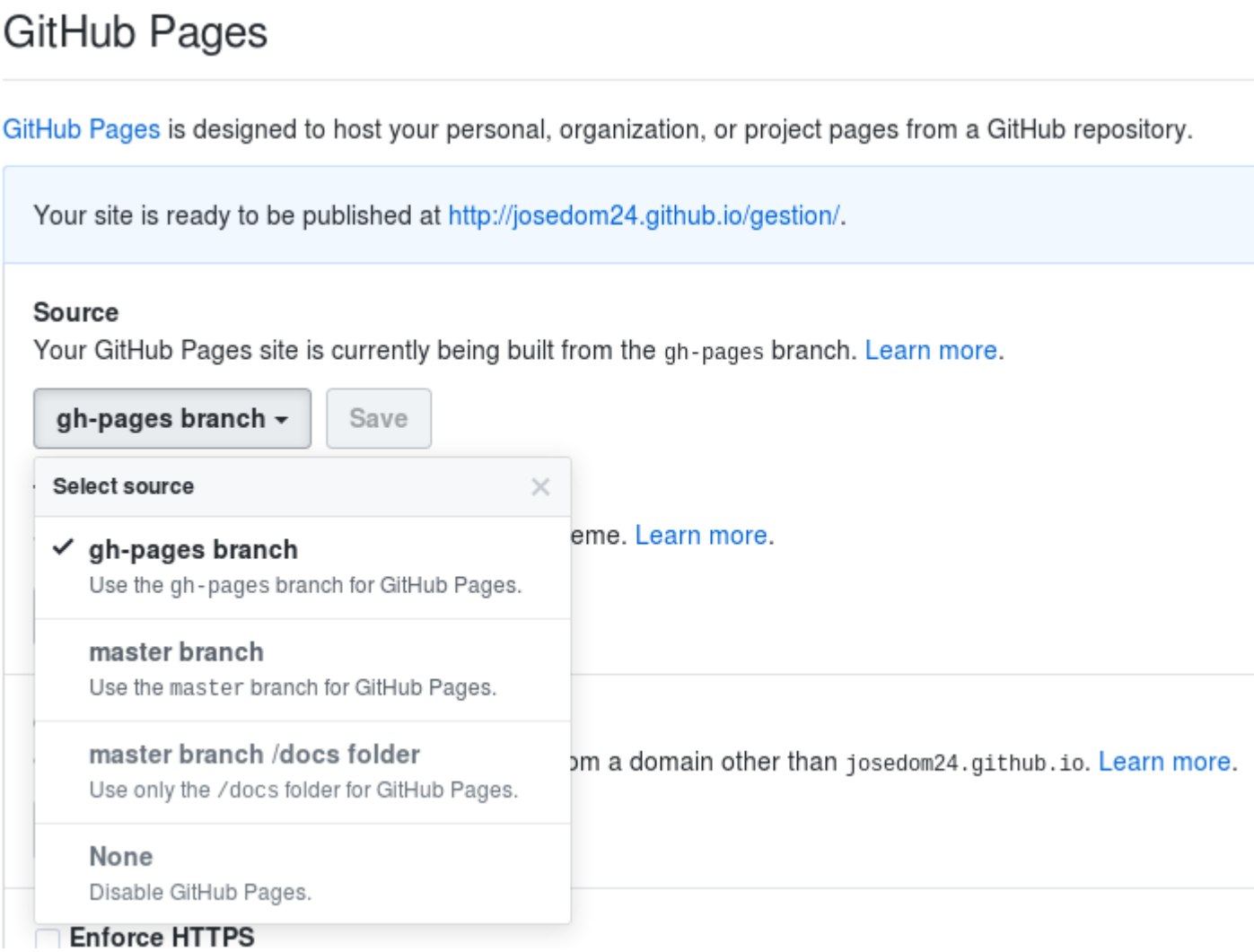
A continuación tenemos que crear la nueva rama:

```
$ cd repository
$ gitcheckout --orphangh-pages
# Createsourbranch, withoutanyparents (it'sanorphan!)
# Switchedto a new branch 'gh-pages'
$ gitrm -rf .
# Removeall files fromtheoldworkingtree
# rm'.gitignore'
```

Para terminar subiendo el primer fichero de nuestra página:

```
$ echo "My GitHub Page" > index.html
$ gitadd index.html
$ gitcommit -a -m "Firstpagescommit"
$ gitpushorigingh-pages
```

Para publicar nuestra página, cómo indicábamos anteriormente, sólo tendríamos que ir a la configuración del repositorio y activar la opción de GitHub Pages seleccionado la rama `gh-page`:

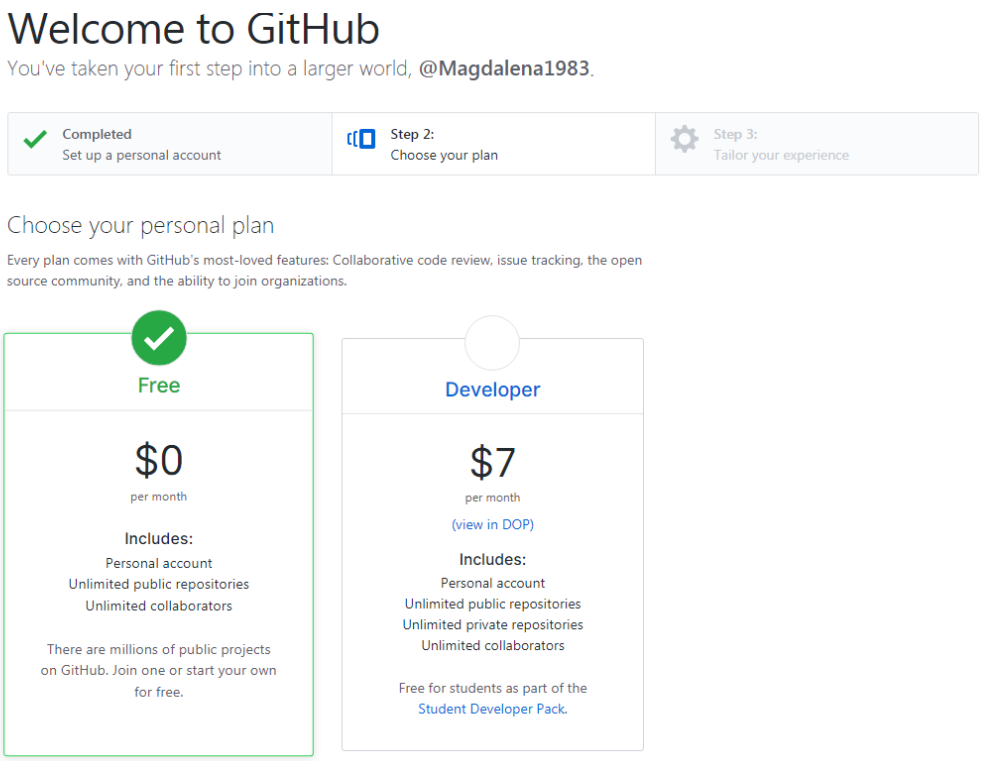


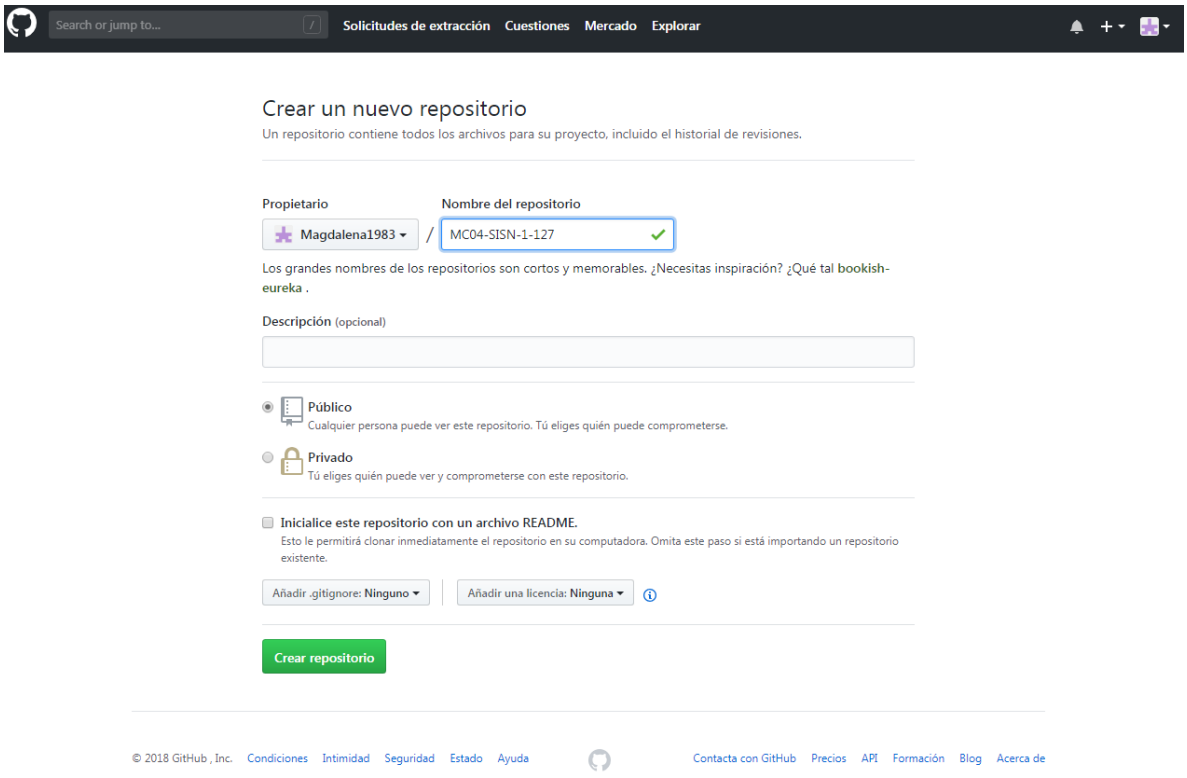
Cómo podemos construir nuestras páginas web

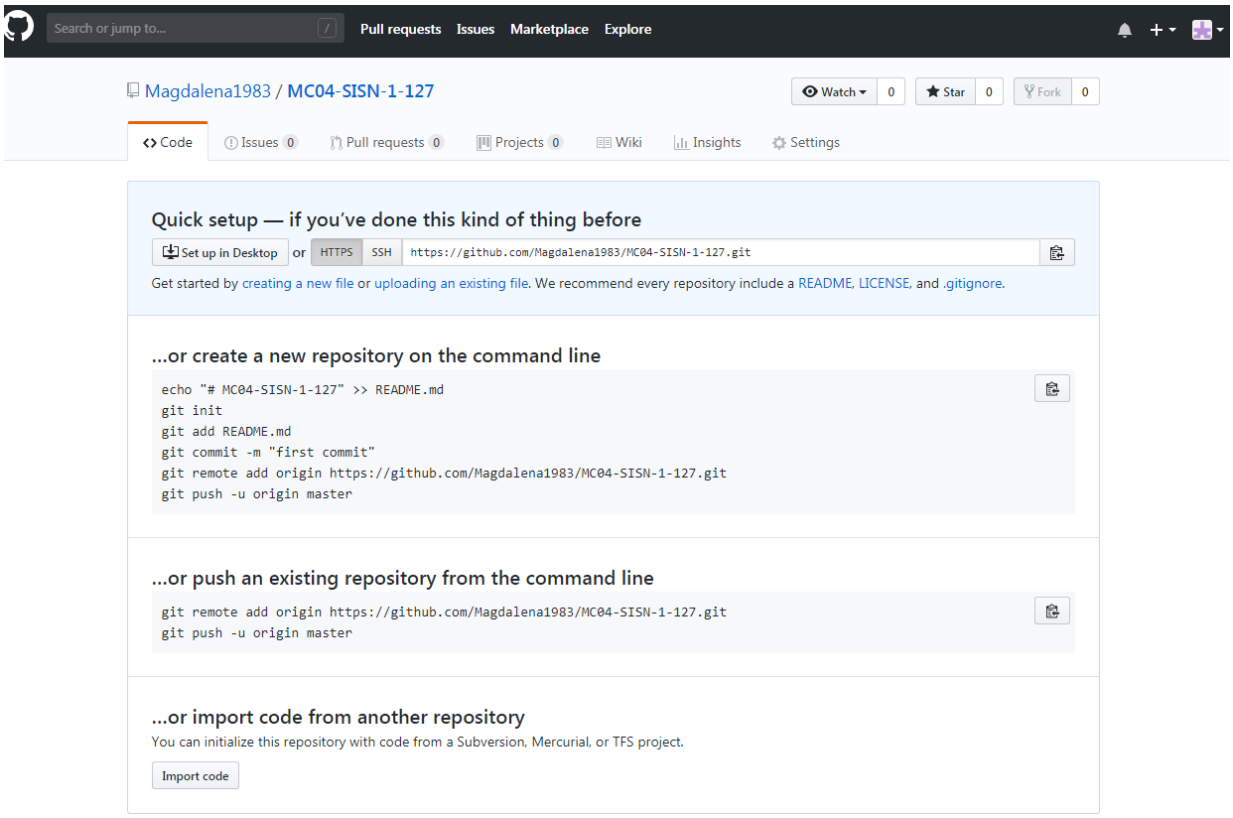
La forma más sencilla de construir nuestro sitio es subir a nuestro repositorio todos los ficheros necesarios: ficheros html, hojas de estilos, javascript, imágenes, etc. Si sólo tuviéramos esta opción de edición de páginas no tendríamos grandes ventajas para decidirnos a escoger este servicio de hosting.

Lo que realmente hace esta herramienta una opción muy potente es que Pages suporta Jekyll, herramienta escrita en Ruby que nos permite generar, de una forma muy sencilla, ficheros HTML estáticos. Aunque esta herramienta esta pensada para generar blogs, nosotros vamos a utilizar algunas de sus funcionalidades para crear páginas estáticas convencionales.

ANEXOS CREACION DE GITHUB




Search or jump to...
Solicitudes de extracción
Cuestiones
Mercado
Explorar
Crear un nuevo repositorio
Un repositorio contiene todos los archivos para su proyecto, incluido el historial de revisiones.
Propietario
Nombre del repositorio
Magdalena1983
MC04-SISN-1-127
Los grandes nombres de los repositorios son cortos y memorables. ¿Necesitas inspiración? ¿Qué tal bookish-eureka .
Descripción (opcional)
Público
Cualquier persona puede ver este repositorio. Tú eliges quién puede comprometerse.
Privado
Tú eliges quién puede ver y comprometerse con este repositorio.
Inicialice este repositorio con un archivo README.
Esto le permitirá clonar inmediatamente el repositorio en su computadora. Omita este paso si está importando un repositorio existente.
Añadir .gitignore: Ninguno
Añadir una licencia: Ninguna
Crear repositorio
© 2018 GitHub , Inc.
Condiciones
Intimidad
Seguridad
Estado
Ayuda
Contacta con GitHub
Precios
API
Formación
Blog
Acerca de


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Magdalena1983 / MC04-SISN-1-127
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
https://github.com/Magdalena1983/MC04-SISN-1-127.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.
...or create a new repository on the command line
echo "# MC04-SISN-1-127" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Magdalena1983/MC04-SISN-1-127.git
git push -u origin master
...or push an existing repository from the command line
git remote add origin https://github.com/Magdalena1983/MC04-SISN-1-127.git
git push -u origin master
...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code