1.7.- Contenido 7: Aplicar procedimiento de publicación de un sitio web utilizando Github Pages para que esté disponible a los visitantes

Objetivo de la jornada

1. Distingue los conceptos fundamentales relacionados con dominio y hosting para una página web.
2. Aplica procedimiento de habilitación de GitHub Pages para el alojamiento de la página web.
3. Aplica procedimiento de subida de cambios a la página web en GitHub Pages.

1.7.1.- Hosting

1.7.1.1. Qué es un hosting.

Hasta el momento hemos abordado un conjunto de tecnologías y herramientas que nos permiten desarrollar nuestros propios sitios web. Hemos desarrollado páginas web simples en base a HTML, CSS, JavaScript y algunas librerías adicionales que nos ofrecen mayores prestaciones. Sin embargo, sólo hemos visto nuestros resultados localmente en nuestro PC de desarrollo. Por lo tanto, debemos llenar un vacío de conocimiento que aún mantenemos en relación a cómo disponibilizar en la world wide web las páginas que desarrollamos para que usuarios de cualquier parte del mundo puedan acceder a ellas. El término comúnmente empleado para referirnos a la prestación que entrega un servidor conectado a la www para disponibilizar al mundo nuestra página es Hosting (Hospedaje), considerando que esa máquina "hospeda" a nuestros documentos o aplicación web para que pueda residir en el "ciberespacio".

SERVIDORES, DIRECCIONES IP Y PUERTOS TCP

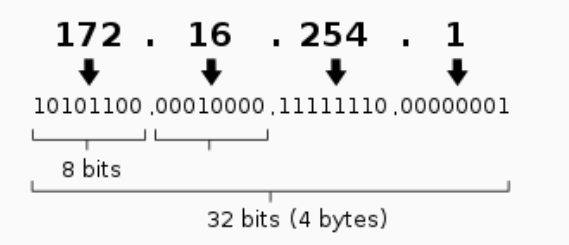
Como vimos en la sección 1.1.1.1., internet está compuesta por una cantidad masiva de servidores que están conectados permanentemente a ella intercambiando datos. Dentro de otros propósitos, la gran mayoría de estos servidores están principalmente dedicados a ser una parte activa de la World Wide Web. Toda esta capa de software que constituye la web, descansa sobre una gran infraestructura física y protocolos de más bajo nivel, compuesta de muchos routers, otras máquinas y servidores de aplicaciones de red, que proveen recursos y lógica para asegurar la comunicación entre nodos remotos.

En sus orígenes la columna vertebral de la internet fueron los miembros de ARPANET. Sin embargo, en la actualidad hay muchos protagonistas involucrados contribuyendo con grandes cantidades de routers de red y servidores: universidades, entidades de gobierno y privadas, además de muchas compañías de telecomunicaciones e ISPs (Internet Service Providers) que son las encargadas de canalizar el acceso de muchos millones personas a esta red pública a nivel global.

Cada uno de estos routers y servidores tiene cabida dentro de la red por contar con una o más direcciones IP públicas, que son identificadores únicos de cada una de sus interfaces de red a través de las cuales se conectan a la internet.

En la frontera entre la esta gran red pública y el usuario final, están los routers que tenemos en nuestros hogares y oficinas, que se encargan de permitir que hayan redes locales en nuestras dependencias y éstas se conecten a la red pública a través de una de sus interfaces de red que posea IP pública. Las direcciones IP que son asignadas a nuestros computadores personales o dispositivos portátiles en redes cableadas o inalámbricas, en residencias u oficinas, son generalmente privadas. Esto significa que la misma dirección asignada a una máquina, puede estar siendo asignada en otra red local, a otra máquina, por un router que cumple esta misma función. Así, generalmente nuestros dispositivos no están expuestos directamente a internet para agentes remotos que quisieran acceder. No es el caso de las máquinas que llamamos servidores y que, particularmente, estén operando como servidores del protocolo HTTP, los que estarán disponibles para atender requerimientos que provengan de cualquier cliente de la internet.

Una dirección IP está conformada de la siguiente manera:



Fuente: Wikipedia - https://en.wikipedia.org/wiki/IP_address

Las direcciones que están en los siguientes rangos corresponden a direcciones locales, y generalmente nuestro PC o dispositivo en que utilicemos nuestro navegador estaremos posicionados en una de ellas:

10.0.0.0 – 10.255.255.255,
172.16.0.0 – 172.31.255.255,
192.168.0.0 – 192.168.255.255

Las direcciones que están fuera de estos rangos, son direcciones públicas, y son las que generalmente poseerán los servidores HTTP que accedamos a través de la internet.

Las direcciones IP en el mundo son administradas por la **ICANN *(Corporación de Internet para la Asignación de Nombres y Números).*** ICANN es una organización que opera a nivel multinacional/internacional. La Autoridad de Números Asignados en Internet (IANA) es un departamento de ICANN que es responsable del mantenimiento de identidades únicas de registradores de internet, que incluyen los nombres de dominios, parámetros de protocolo, direcciones IP y Sistemas Autónomos de Números.

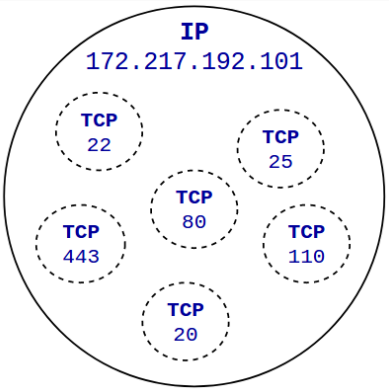
Un servidor expuesto a internet está permanentemente ejecutando una infinidad de procesos, y muchos de ellos tienen conexión a la red para diversos objetivos: Intercambio de información entre bases de datos residentes en los servidores, acceso de usuarios administradores, intercambio de archivos de usuario, intercambio de datos en tiempo real con hardware remoto de diverso tipo, y muchos más. De esta forma, para acceder a un recurso web como los descritos en la sección 1.1.1.1., no basta con indicar la dirección IP del servidor que está conectado a la red pública (internet). Al recibir un requerimiento externo, cada servidor necesita conocer a qué proceso o servicio interno va dirigida la petición de conexión. De esta forma surge la necesidad de manejar el concepto protocolo TCP, que opera en un nivel jerárquico más alto que el protocolo IP.

El protocolo TCP garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir los distintos procesos o aplicaciones que están expuestos a la red dentro una misma máquina. Para esto se incorpora el concepto de puerto TCP. Como consecuencia de esto, tenemos la posibilidad de establecer una multiplicidad de conexiones a esa máquina a través de distintos puertos en la misma dirección IP. Cada una de estas conexiones se lleva a cabo a través de un identificador único que se define por la combinación de la dirección IP de la máquina y el puerto TCP del proceso específico que está comunicando datos y que se denomina Socket:

```
Socket = Dirección-IP:Puerto-TCP
```

De esta forma, podemos imaginar la dirección IP como una troncal de acceso a la comunicación con un servidor remoto, y los distintos puertos como canales a través de los cuáles accederemos a las diferentes aplicaciones o procesos:

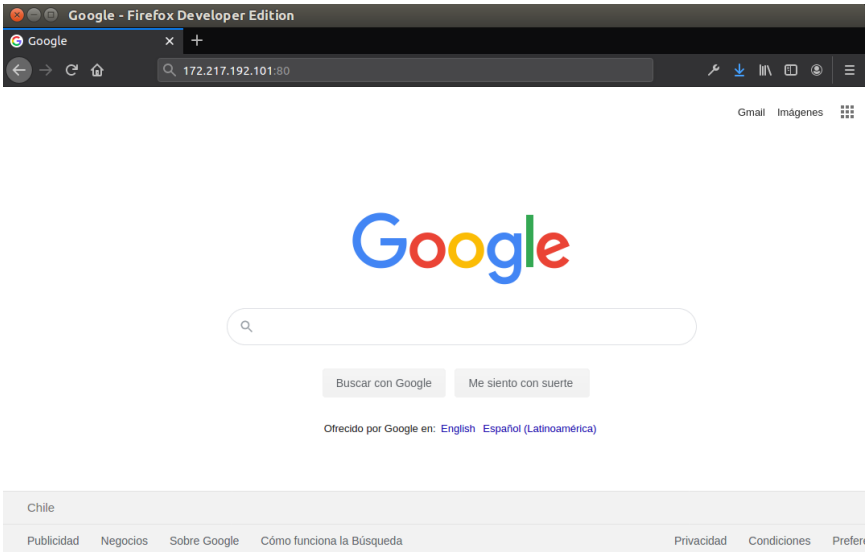
Ejemplo de un servidor con IP 172.217.192.101 con servicios típicos



Servicio	Socket
SSH	172.217.192.101:22
HTTP	172.217.192.101:80
SMTP	172.217.192.101:25
POP3	172.217.192.101:110
Telnet	172.217.192.101:20
HTTPS	172.217.192.101:443

TCP da soporte a muchas de las aplicaciones más populares de Internet (navegadores, intercambio de ficheros, clientes FTP, etc.) y protocolos de aplicación HTTP, SMTP, SSH y FTP, que operan en determinados puertos predefinidos. Para el caso particular de HTTP, el servidor tiene un proceso o aplicación llamado Servidor HTTP, que está permanentemente "escuchando" el puerto TCP 80, esperando requerimientos para atenderlos, ya sea redirigiendo la solicitud o entregando los recursos web solicitados.

De esta manera, y para tangibilizar más estos últimos conceptos, si entramos a nuestro browser y colocamos en la barra de dirección el identificador de socket **172.217.192.101:80**, correspondiente a HTTP en de la última figura, obtendremos lo siguiente:

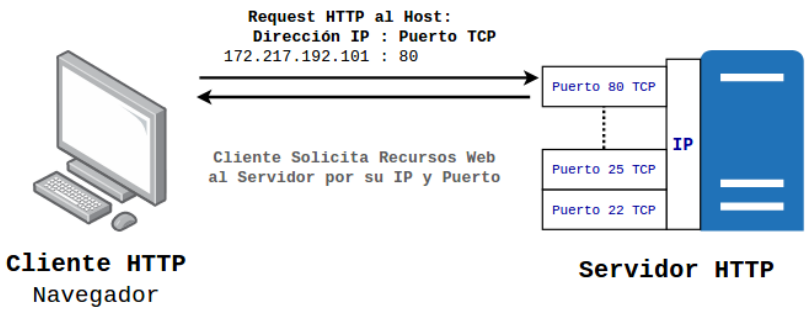


Podemos ver que el servidor con la dirección IP **172.217.192.101**, resulta ser google y accediendo al puerto 80 obtenemos la página HTML que sus desarrolladores han puesto a disposición de todos los usuarios de navegadores de internet.

El puerto 80 fue escogido por Tim Berners-Lee en 1991 como el puerto por defecto para acceder a los contenidos web HTML, siempre que no se indicara otro. Por lo tanto, en los navegadores no es necesario especificar que deseamos conectarnos al puerto 80 sino que éste lo dará por hecho al escribir una dirección que inicie con **http**, tal como http://www.google.com. En muchos casos, en la actualidad, los servidores redirigen nuestros requerimientos **http** de puerto **80**, a su equivalente **https** (Protocolo HTTP encriptado) al puerto **443**. Adicionalmente los navegadores soportan otros protocolos, como por ejemplo **ftp** para transferencia de archivos o su equivalente encriptado **ftps**.

DOMINIOS Y DNS

Hasta este punto hemos descrito cómo el navegador apunta en internet hacia el servidor de destino indicado y se conecta a él por el puerto 80, donde existe un proceso que está permanentemente "escuchando" para proveer los servicios solicitados.



Solicitud del navegador al servidor por Dirección IP y Puerto TCP.

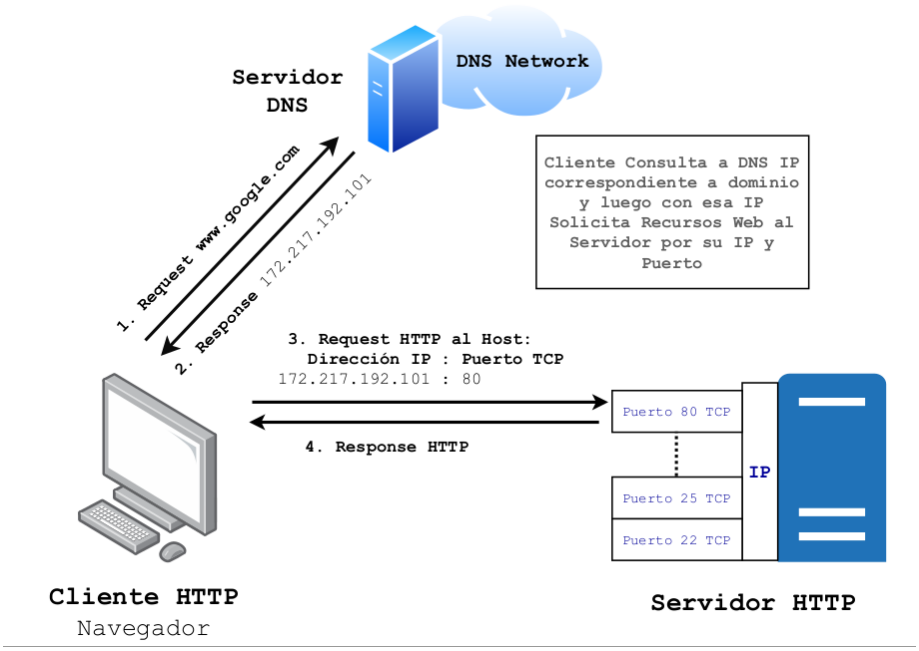
Sin embargo, normalmente como usuarios y visitantes de sitios web residentes en diversos servidores existentes en la web no utilizamos direcciones IP para referirnos al sitio que queremos visitar. Normalmente utilizamos un nombre alfanumérico que nos resulta más fácil recordar y que se denomina **nombre de dominio**.

Al igual que la administración de direcciones IP, la organización IANA/ICANN administra los nombres de dominio, tales como google.com, yahoo.com, facebook.com, twitter.com, y muchos otros. Al comprar un dominio, el propietario de éste debe con un servicio que le permita registrar la relación entre dominio y dirección IP del servidor que vaya a utilizar. Posteriormente, esa relación entre dominio y dirección IP debería ser difundida a todos los navegadores del mundo para que éstos sepan que al recibir una solicitud del usuario de acudir a ese dominio, éstos deben apuntar a la dirección IP que ha sido configurada. Para esto existe una red de servidores a nivel mundial, que poseen un proceso o aplicación expuesta a internet a “escuchando” el puerto **TCP 53**, a través del cual reciben requerimientos de traducción de dominio a dirección IP, y se denominan DNS.

El DNS (Sistema de nombres de dominio) es un sistema de nombres jerárquico y descentralizado para recursos conectados a Internet. El DNS mantiene una lista de nombres de dominio junto con los recursos, como las direcciones IP, que están asociados con ellos.

La función más destacada del DNS es la traducción de nombres de dominio amigables para el ser humano a una dirección IP numérica. Este proceso de mapear un nombre de dominio a la dirección IP apropiada se conoce como una búsqueda de DNS. Por el contrario, se utiliza una búsqueda DNS inversa (rDNS) para determinar el nombre de dominio asociado con una dirección IP.

Es usual que el sistema operativo de nuestros dispositivos, al momento de conectarse a una red local y obtener una dirección IP asignada por el router, reciba también la dirección de DNS promovida por éste. Por ejemplo, es muy usual utilizar las direcciones IP DNS de google (8.8.8.8 y 4.4.4.4) o de otros proveedores de este servicio con características más específicas como control parental, tal como OpenDNS (208.67.222.123 y 208.67.220.123). También es posible configurar forzadamente en el nuestro PC el uso de un DNS específico.



Solicitud del navegador al servidor por Dirección IP y Puerto TCP.

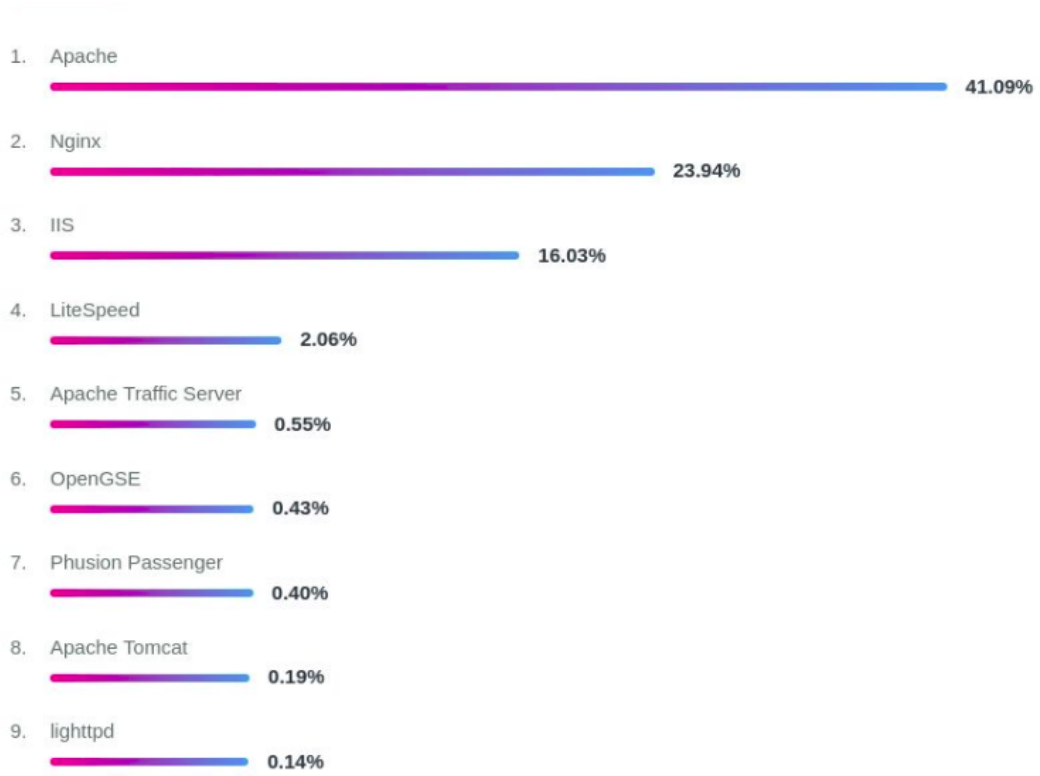
EL SERVIDOR HTTP

Hasta el momento hemos comprendido cómo el navegador de los visitantes de nuestros sitios, o aplicaciones web, consiguen llegar a la máquina o servidor de destino donde tenemos “alojado” nuestro contenido. Hemos mencionado que en el servidor el puerto TCP 80 está permanentemente siendo escuchado por un proceso o aplicación llamado Servidor HTTP. Éste corresponde a un software que ha sido desarrollado para “Servir” contenidos web, tanto estáticos como dinámicos.

- Un **servidor web estático**, consiste en una máquina con un servidor HTTP (software). Llamamos a este "estático" debido a que el servidor envía los archivos tal como están almacenados al cliente/navegador como contenido de la respuesta HTTP.

- Un **servidor web dinámico** consiste en un servidor web estático con un software extra, que en general debe ser un servidor de aplicación y una base de datos. Llamamos a esto "dinámico" porque el servidor de aplicación actualiza los archivos almacenados en función de ciertos criterio e interactuando con una base de datos antes de enviarlos mediante el servidor HTTP.

En la actualidad existe una gran variedad de Servidores HTTP, siendo los más comunes: Apache HTTP Server, NGINX, Apache Tomcat, Node.js, Lighttpd y otros. Las cuotas de mercado de los distintos Servidores HTTP del mercado son aproximadamente como se muestra a continuación:



Fuente (2020): https://es.hostadvice.com/marketshare/server/

HOSTING EN LA PRÁCTICA

Para aplicar todo lo anterior a un caso práctico general, tendremos que completar los siguientes pasos:

1. Completar nuestro desarrollo y tenerlo listo para ponerlo en producción, con un repositorio git, en GitHub, Bitbucket, GitLab u otro.
2. Comprar un dominio para nuestra aplicación, u obtener uno sin costo en algunos de los proveedores existentes.

3. Contar con un servicio de DNS que nos permita propagar la información de la relación dominio/dirección IP en la internet.
4. Obtener un servidor físico o virtual, con una dirección IP Pública asignada, que pueda “alojar” o “hostear” nuestros recursos. Normalmente este servidor operará en base a alguna distribución de GNU/Linux.
5. Instalar un software de servidor HTTP que cumpla con nuestras necesidades. Típicamente Apache HTTP Server podrá ser una buena opción.

En nuestro caso recurriremos a una solución (GitHub Pages) que provee un atajo para hacer una publicación de nuestro trabajo prescindiendo de varios de los pasos descritos arriba y ahorrando gastos que, para nuestra etapa de aprendizaje, son innecesarios.
A pesar de utilizar GitHub Pages para propósitos de aprendizaje, recomendamos investigar más sobre la metodología recién descrita, que será la que deberemos utilizar al momento de desplegar en producción un desarrollo para un cliente o para un proyecto que queramos realizar con propósitos más amplios.

1.7.2.- GitHub Pages

Luego de haber revisado todos los conceptos de cómo nuestros desarrollos web pueden quedar disponibilizados en la world wide web, veremos una opción de cómo publicar fácilmente, de forma muy visual e intuitiva, o bien por medio de comandos, tu sitio a través de GitHub Pages, un servicio gratuito provisto por GitHub.

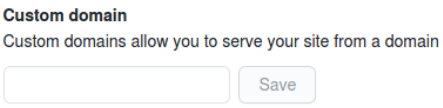
GitHub Pages es un servicio de alojamiento de sitio estático que toma archivos HTML, CSS y JavaScript directamente desde un repositorio en GitHub, opcionalmente ejecuta los archivos a través de un proceso de compilación y publica un sitio web. De esta forma, este servicio será de utilidad en casos que tengamos sitios webs o páginas que no requieran de la operación de un servidor de aplicación o base de datos[2].

VENTAJAS DE GITHUB PAGES
Si limitamos nuestro análisis a sitios que sólo requieren hosting estático para entrega de archivos HTML, CSS y JavaScript, entonces podemos decir que GitHub Pages nos entrega las siguiente ventajas:

1. **No tiene costo:** En etapas de aprendizaje y cuando estamos en un contexto de práctica y experimentación en el desarrollo de sitios web estáticos, es de gran utilidad contar con un servicio en el cual no debamos incurrir en costos,

y que nos entregue facilidades equivalentes a tener espacio en un servidor, un dominio, no necesitar configuraciones de DNS, y no tener que preocuparnos por el mantenimiento de la plataforma de un servidor (Sistema Operativo, Servidor HTTP, etc.).

2. **Dominio Personalizado:** GitHub Pages provee la posibilidad de configurar un dominio alternativo a la URL por defecto del tipo https://usuario.github.io/nombre-repositorio.



3. **Provee HTTPS**: Nos ahorramos la necesidad de instalar certificados SSL y el trabajo asociado a tener un sitio con protocolo HTTPS. En caso de utilizar la URL por defecto asignada por GitHub Pages la opción de HTTPS es mandatoria. En caso de dominio personalizado, esto es opcional.

4. **Conexión Directa al Repositorio:** GitHub Pages aprovecha al máximo el hecho de tener el repositorio en su propia infraestructura. De esta manera evitamos estar sincronizando un servidor con el repositorio remoto cada vez que queramos realizar despliegue de actualizaciones de nuestro sitio. GitHub Pages puede hacer visible en producción la rama **máster** o la rama **gh-pages**, entregando la flexibilidad de no estar forzado a desplegar la rama máster. En caso de repositorios privados, esto permite escoger cuál es la rama del código que queda abierta al público.

5. **Colaboración Directa:** Por estar conectado directamente el sitio en producción al repositorio, es muy fácil recibir pull requests o reporte de issues, de parte de otros desarrolladores o usuarios, para mejoras del sitio o aplicación que estemos publicando con GitHub Pages.

6. **Integración directa con Jekyll**: GitHub Pages provee el sistema Jekyll incorporado por defecto, en caso de necesitar comenzar con presencia en la web antes de tener un desarrollo ya terminado. Jekyll es un generador simple para sitios web estáticos con capacidades de blog; adecuado para sitios web personales, de proyecto o de organizaciones, incluso para usuarios sin conocimientos de desarrollo.

7. **Múltiples repositorios publicados:** Permite la publicación de múltiples proyectos, alojados en diferentes repositorios, con una URL distinta y posible de personalizar con nombres de dominio.

CÓMO HABILITAR GITHUB PAGES

La habilitación de un proyecto web estático en GitHub Pages es realmente fácil. Debemos contar con un proyecto ya en estado de poder ser publicado.

A modo de ejemplo, recurriremos al proyecto realizado en la sección 1.4.1.3. que utilizamos a modo de demostración de Bootstrap. Nuestro proyecto tiene la siguiente estructura de archivos:

```
.
├── css
│   └── style.css
├── img
│   ├── bg-wind-generators.jpg
│   ├── flag-64.png
│   ├── gear-64.png
│   ├── group1.jpg
│   ├── group2.jpg
│   ├── group3.jpg
│   ├── screen-64.png
│   └── world-64.png
└── index.html
```

Debemos inicializar un repositorio git dentro de la carpeta raíz de nuestro proyecto:

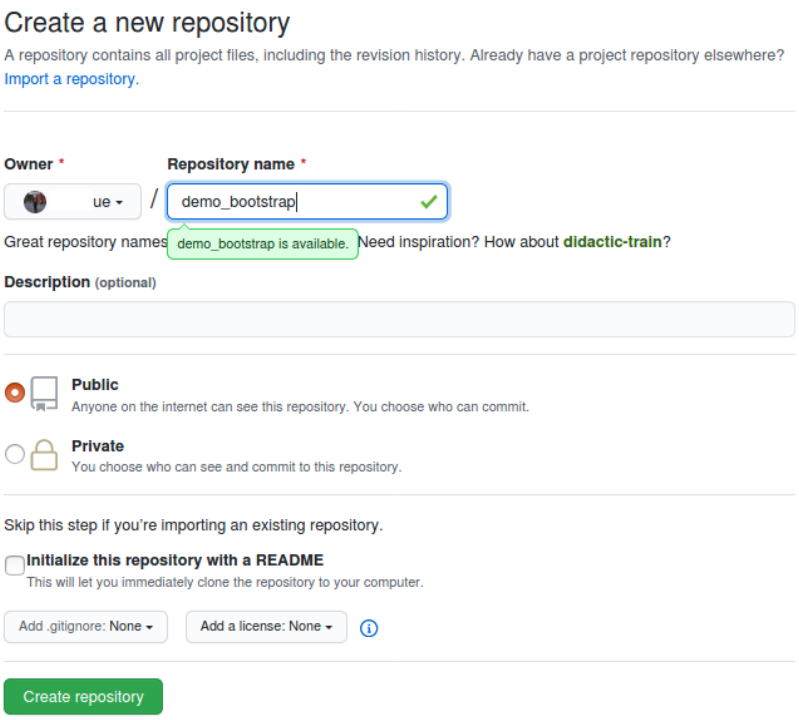
```
user@machine:~/.../github_pages/demo_bootstrap$ git init
Initialized empty Git repository in
/home/user/Documents/awakelab/full_stack_python/github_pages/de
mo_bootstrap/.git/

user@machine:~/.../github_pages/demo_bootstrap$
```

Luego hacemos add y commit de todos los archivos del proyecto tal como lo terminamos en la sección 1.4.1.3.

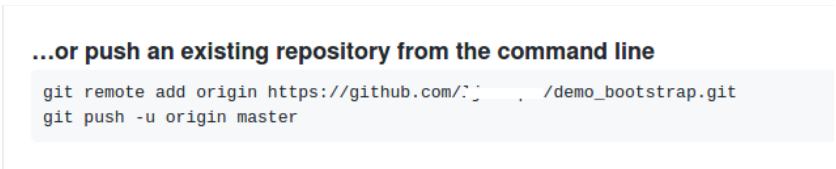
```
user@machine:~/.../github_pages/demo_bootstrap$ git add --all ; git commit
-m "commit inicial"
[master (root-commit) 5012a83] commit inicial
10 files changed, 522 insertions(+)
create mode 100644 css/style.css
create mode 100644 img/bg-wind-generators.jpg
create mode 100644 img/flag-64.png
create mode 100644 img/gear-64.png
```

```
create mode 100644 img/group1.jpg
create mode 100644 img/group2.jpg
create mode 100644 img/group3.jpg
create mode 100644 img/screen-64.png
create mode 100644 img/world-64.png
create mode 100644 index.html
```

Luego nos dirigimos a nuestra cuenta de GitHub y creamos un nuevo repositorio al que llamaremos **demo_bootstrap**:



A continuación seguimos las instrucciones para hacer **push** de nuestro repositorio existente:



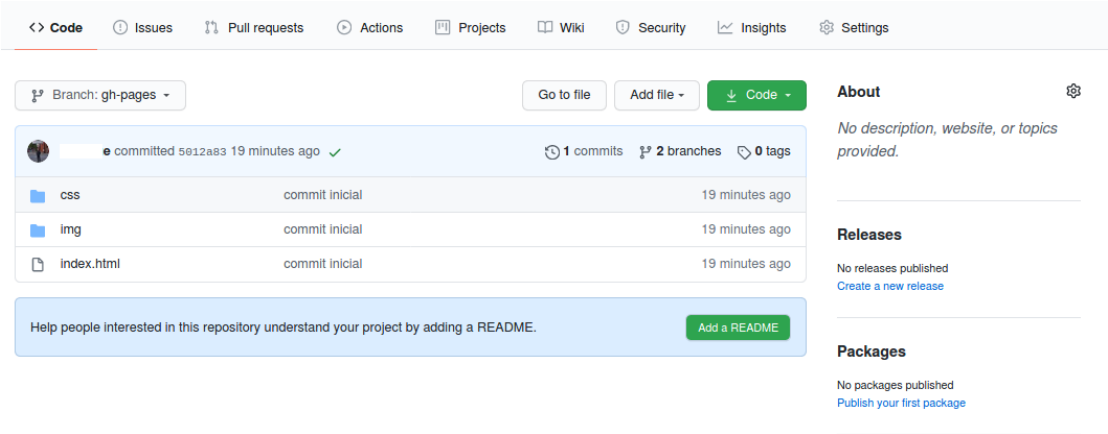
Si el push concluye existosamente entonces, creamos una nueva rama llamada **gh-pages**:

```
user@machine:~/.../github_pages/demo_bootstrap$
git checkout -b gh-pages
Switched to a new branch 'gh-pages'
```

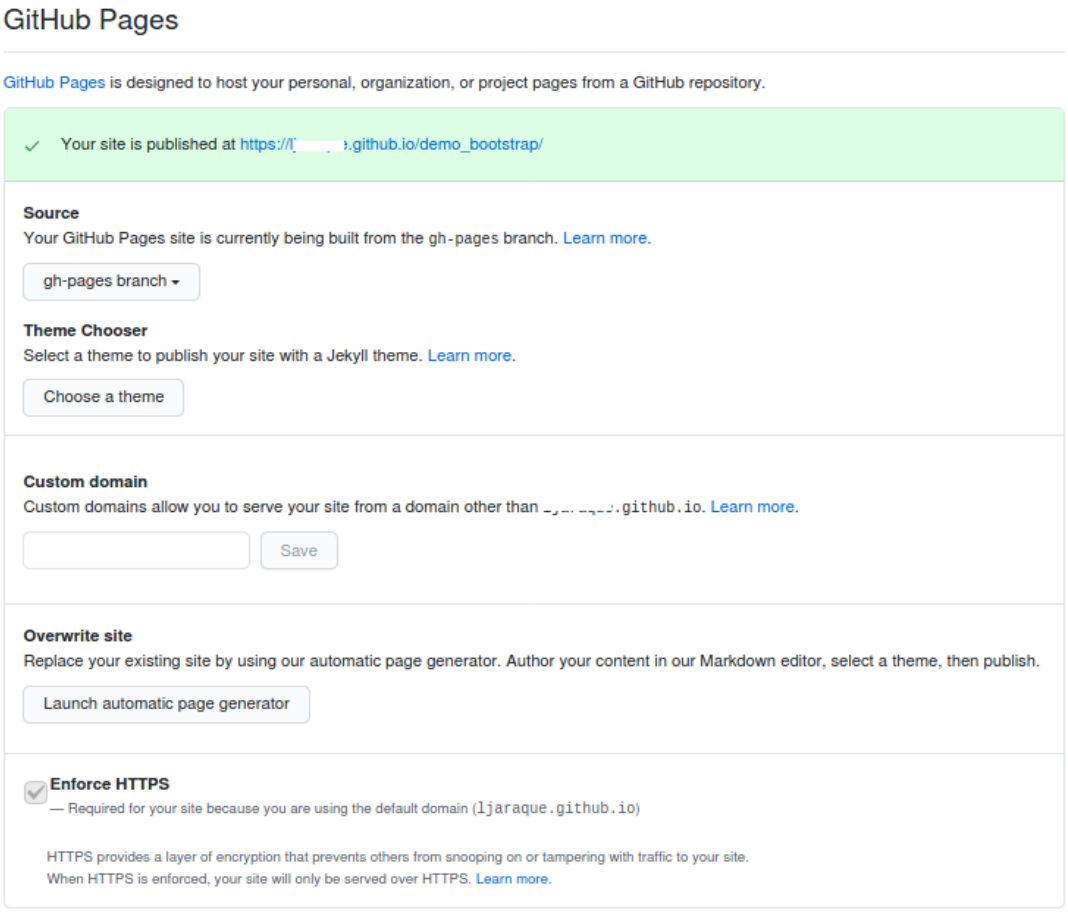
Luego hacemos push de esta nueva rama:

```
git push origin gh-pages
```

De ser exitoso el push de la rama **gh-pages** entonces ya la tendremos en el repositorio.



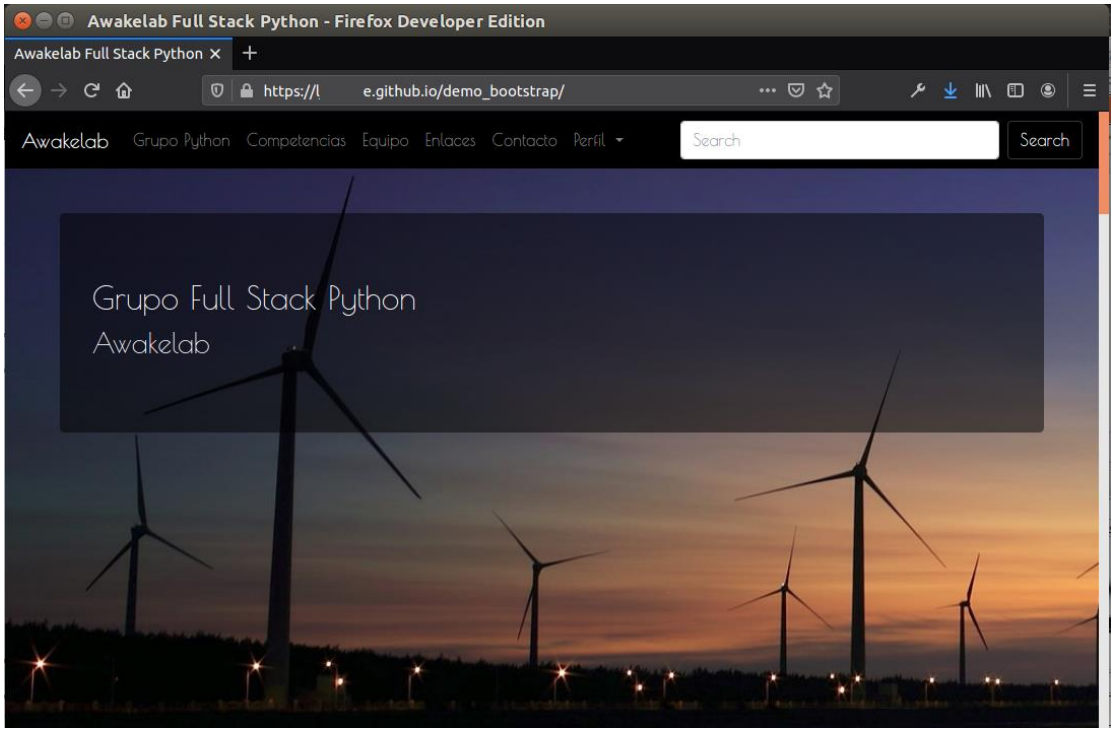
Si vamos al menú **Settings**, encontraremos las opciones relacionadas con GitHub Pages, para las configuraciones del servicio de hosting del repositorio **demo_bootstrap**.



La configuración por defecto define que la rama a ser mostrada en nuestra URL **https://user.github.io/demo_bootstrap/** es **gh-pages**, que hemos subido en nuestra última acción de git.

Nota: Es importante no olvidar el símbolo "/" al final de la URL al momento de ingresar la dirección en nuestro navegador.

De esta forma, al ir en nuestro navegador a la dirección mencionada veremos publicado al mundo nuestro proyecto de demostración que anteriormente sólo podíamos ver en nuestro PC de desarrollo o desde otro dispositivo dentro de la misma red local, usando la extensión Live Server de Visual Studio Code. El resultado es el siguiente:



1.7.3.- Referencias

[1] IANA
https://es.icannwiki.org/IANA

[2] Getting Started with GitHub Pages
https://guides.github.com/features/pages/