

Jekyll

Jekyll

Jekyll es un tipo de **CMS** especial. Pertenece a la categoría de Headless CMS también llamados **Static Site Generator**.

En este tipo de **CMS** no necesitamos un lenguaje de programación en la parte del servidor, porque todo el contenido es estático. Ahora bien, si el contenido es estático, ¿dónde está la ventaja?

Pues la ventaja viene porque estos **CMS** generan contenido estático a partir de ficheros fuente, sin necesidad de tener ni tan siquiera una base de datos.

Este contenido estático, pero generado dinámicamente, es el que compone nuestra aplicación.

Cuáles son las ventajas

- Menos complejos
- Mayor velocidad
- Mayor seguridad
- Mayor escalabilidad
- Control de versiones

Top ten

- <https://www.creativebloq.com/features/10-best-static-site-generators>
- <https://www.netlify.com/blog/2017/05/25/top-ten-static-site-generators-of-2017/>

Más información acerca de **Headless CMS**

- <https://jamstack.org/>
- <https://www.staticgen.com/>

Para nuestra página personal de **GitHub** vamos a usar Jekyll, pues es mantenido por el propio **GitHub** y es soportado nativamente.

Instalación

Jekyll está basado en Ruby, por lo que el primer paso es instalar **Ruby**

```
sudo apt-get install ruby-full build-essential zlib1g-dev
```

Después hay que configurar nuestras variables de entorno para no tener que usar **sudo** al instalar **gemas** (las gemas son como los paquetes de **composer** de **PHP** o **npm** de **node.js**)

```
echo '# Install Ruby Gems to ~/gems' >> ~/.bashrc
echo 'export GEM_HOME="$HOME/gems"' >> ~/.bashrc
echo 'export PATH="$HOME/gems/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

Y finalmente instalar, **Jekyll**

```
gem install jekyll bundler
```

Más información en:

<https://jekyllrb.com/docs/installation/ubuntu/>

Primera aplicación

Primero vamos a clonar nuestro repositorio personal de **GitHub** en un directorio local.

```
mkdir tu-usuario.github.io.git
cd tu-usuario.github.io.git
git clone https://github.com/tu-usuario/tu-usuario.github.io.git .
```

Y creamos nuestra primera página:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Home</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

Build

Jekyll es un generador de sitios estáticos, por lo que necesitamos que Jekyll construya el sitio antes de poder verlo. Hay dos comandos que puedes ejecutar en la raíz de tu sitio para construirlo:

- `jekyll build`: crea el sitio y genera un sitio estático en un directorio llamado `_site`.
- `jekyll serve --livereload`: hace lo mismo, excepto que se reconstruye cada vez que se realiza un cambio y ejecuta un servidor web local en `http://localhost:4000`.

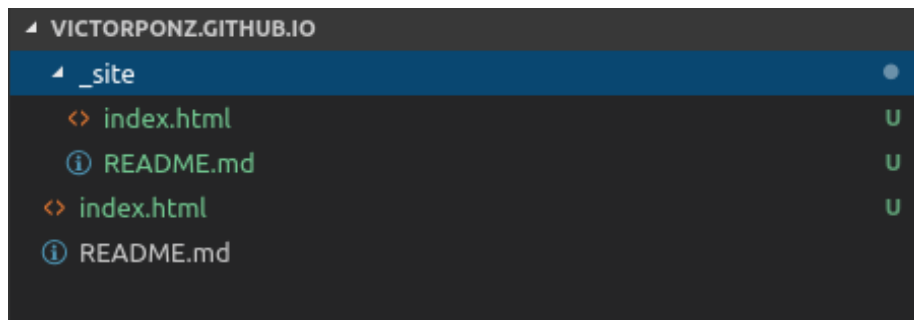


Figure 1: Estructura Directorios Jekyll

Parece que no hace mucho, ya que sólo ha copiado los archivos al directorio `_site`, pero ya veremos que puede hacer muchas más cosas.

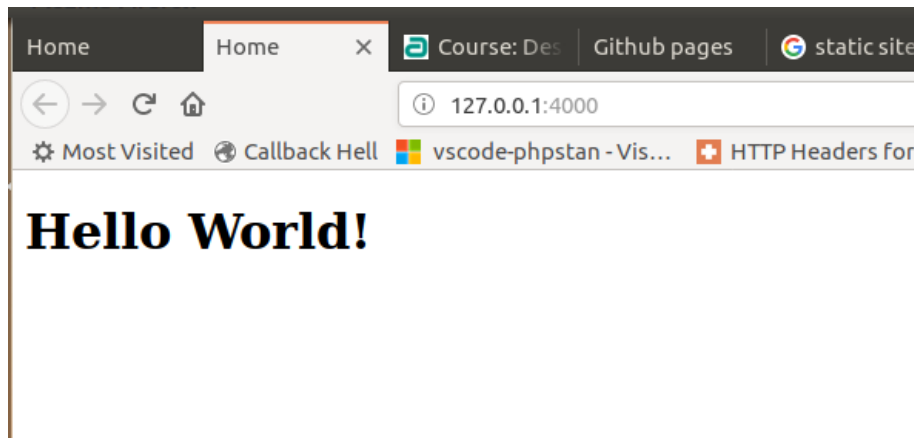


Figure 2: Hello World Jekyll

Como queremos actualizar nuestra página personal, vamos a hacerlo directamente con **git**. En el caso de subir la web a **GitHub**, hemos de ignorar el directorio `_site`, mediante `.gitignore` ya que es el propio **GitHub** quien generará automáticamente nuestra web.

Y desplegamos nuestra primera versión haciendo un **push**.

```
git add .gitignore
git add index.html
git commit -m "jeekyll"
git push
```

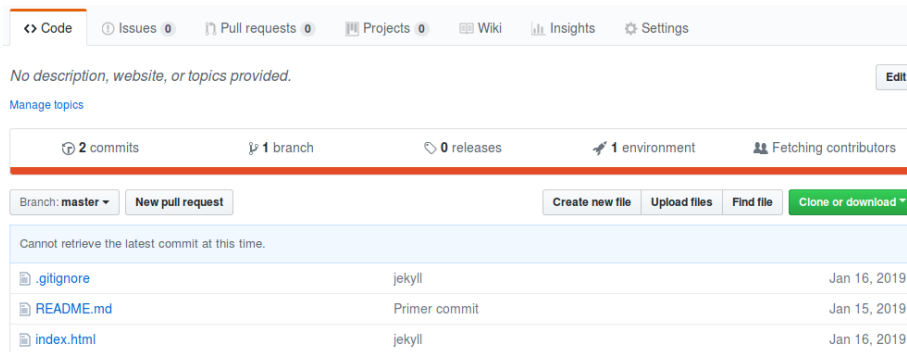


Figure 3: Github Web Pages

Y ahora visitamos <https://victorponz.github.io/>

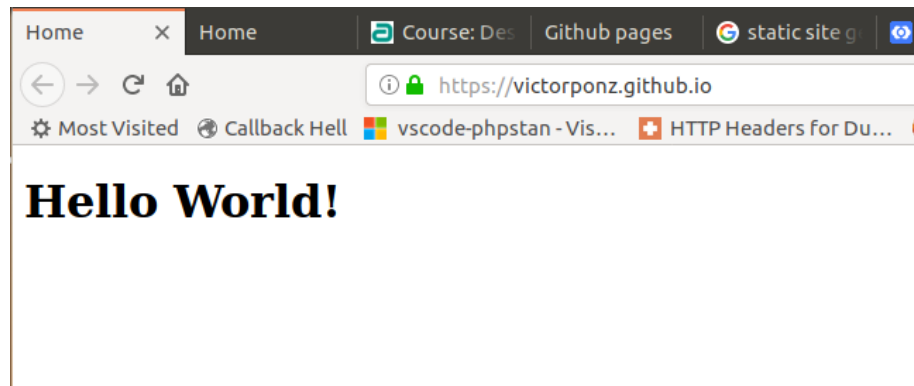


Figure 4: victorponz.github.io

GitHub se ha encargado de todo y ha generado la página personal.

Liquid

Liquid es donde **Jekyll** comienza a ponerse más interesante. **Liquid** es un lenguaje de plantillas que tiene tres partes principales: objetos, etiquetas y filtros.

En esta página tenéis información de cómo usar Liquid en Jekyll.

Ahora cambia `index.html`

```
{% raw %}

---
---
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Home</title>
  </head>
  <body>
    <h1>{{ "Hello World!" | downcase }}</h1>
  </body>
</html>

{% endraw %}
```

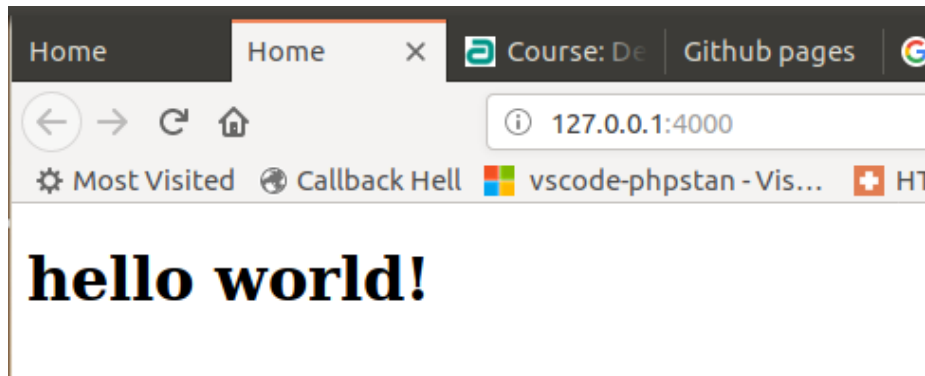


Figure 5: Liquid

Hemos usado el primer objeto, la parte que va entre `{{}}` de Liquid y el primer filtro: `downcase`

Front Matter

Front Matter es un fragmento de YAML que se encuentra entre dos líneas de tres puntos en la parte superior de un archivo. Front Matter se utiliza para establecer variables para la página, por ejemplo:

```
{% raw %}

---
my_number: 5
---
```

```
{% endraw %}
```

Estas variables son accesibles en **Liquid** mediante la variable `page`

Por ejemplo, para imprimir esta variable en una plantilla:

```
{% raw %}

{{ page.my_number }}
```

```
{% endraw %}
```

Vamos a usar una variable `title` en nuestro Front matter

```
{% raw %}

---
title: Home
---
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>{{ page.title }}</title>
  </head>
  <body>
    <h1>{{ "Hello World!" | downcase }}</h1>
  </body>
</html>
```

```
{% endraw %}
```

De momento, tampoco ha cambiado nada. Luego le daremos más funcionalidad.

Layouts

Si queremos hacer una nueva página, por ejemplo `about.html` podríamos copiar `index.html` y modificarlo.

```
{% raw %}

---
title: About
---
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>{{ page.title }}</title>
  </head>
  <body>
    <h1>About page</h1>
    <p>This page tells you a little bit about me.</p>
  </body>
</html>

{% endraw %}
```

Pero claro, ¿dónde está la ventaja de usar **Jekyll** si todo lo hacemos de esta forma?

La respuesta está en los **Layouts**.

Primero, creamos un directorio llamado `_layouts` y creamos un nuevo archivo llamado `default.html` con el siguiente código:

```
{% raw %}

<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>{{ page.title }}</title>
  </head>
  <body>
    {{ content }}
  </body>
</html>

{% endraw %}
```

Es parecido al código a anterior, pero no tiene `Front matter` y el contenido de la página está reemplazado con la variable **Liquid content**

Para que `index.html` use este `layout`, reemplaza todo el contenido por

```
{% raw %}

---
layout: default
title: Home
---
<h1>{{ "Hello World!" | downcase }}</h1>
```

```
{% endraw %}
```

Y reemplaza el código de `about.html`:

```
{% raw %}

---
layout: default
title: About
---
<h1>About page</h1>
<p>This page tells you a little bit about me.</p>

{% endraw %}
```

Includes

Ahora tenemos dos páginas pero no hay forma de navegar entre ellas.

Para ello usamos el **tag include** y colocamos el código en un archivo dentro del directorio `_includes`

Creamos nuestro primer `include` en `_includes/navigation.html`

```
<nav>
  <a href="/">Home</a>
  <a href="/about.html">About</a>
</nav>
```

Y ahora lo usamos en `default.html`

```
{% raw %}
```



```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>{{ page.title }}</title>
  </head>
  <body>
    {% include navigation.html %}
    {{ content }}
  </body>
</html>
```

```
{% endraw %}
```

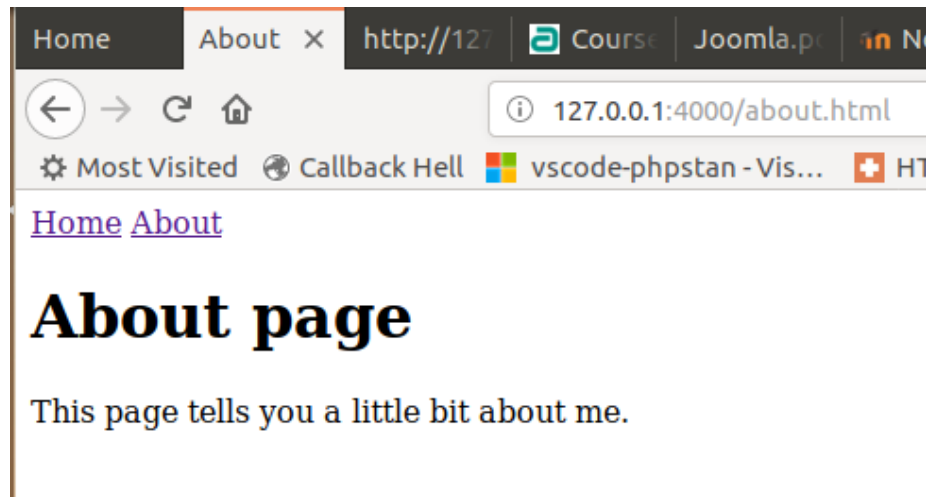


Figure 6: navigation

Resaltar la página actual

Usamos una de las variables que proporciona Jekyll automáticamente: `page.url`

```
{% raw %}
```

```
<nav>
  <a href="/" {% if page.url == "/" %}style="color: red;{% endif %}>
    Home
  </a>
  <a href="/about.html" {% if page.url == "/about.html" %}style="color: red;{% endif %}>
    About
  </a>
</nav>
```

```
{% endraw %}
```



[Home](#) [About](#)

About page

This page tells you a little bit about me.

Todavía hay mucha repetición aquí si desea agregar un nuevo elemento a la navegación o cambiar el color de resaltado. En el siguiente paso abordaremos esto.

Data Files

Jekyll admite la carga de datos desde archivos YAML, JSON y CSV ubicados en un directorio `_data`. Los archivos de datos son una excelente manera de separar el contenido del código fuente para hacer que el sitio sea más fácil de mantener.

En este paso, almacenaremos el contenido de la navegación en un archivo de datos y luego iteraremos sobre él en el include de navegación.

YAML es el formato que más común en el ecosistema de Ruby. Lo usaremos para almacenar una serie de elementos de navegación, cada uno con un nombre y un enlace.

Crea un archivo de datos para la navegación en `_data/navigation.yml` con lo siguiente:

```
- name: Home
  link: /
- name: About
  link: /about.html
```

Jekyll pone este archivo de datos a tu disposición en la variable `site.data.navigation`. En lugar de mostrar cada enlace en `_includes/navigation.html`, ahora podemos iterar sobre el archivo de datos:

```
{% raw %}
```

```
<nav>
  {% for item in site.data.navigation %}
    <a href="{{ item.link }}" {% if page.url == item.link %}style="color: red;"{% endif %}>
      {{ item.name }}
    </a>
  {% endfor %}
</nav>

{% enddraw %}
```

Si ahora añadimos una nueva entrada en `navigation.yml`, automáticamente creará el nuevo enlace:

```
- name: Home
  link: /
- name: About
  link: /about.html
- name: Nueva
  link: /nueva.html
```

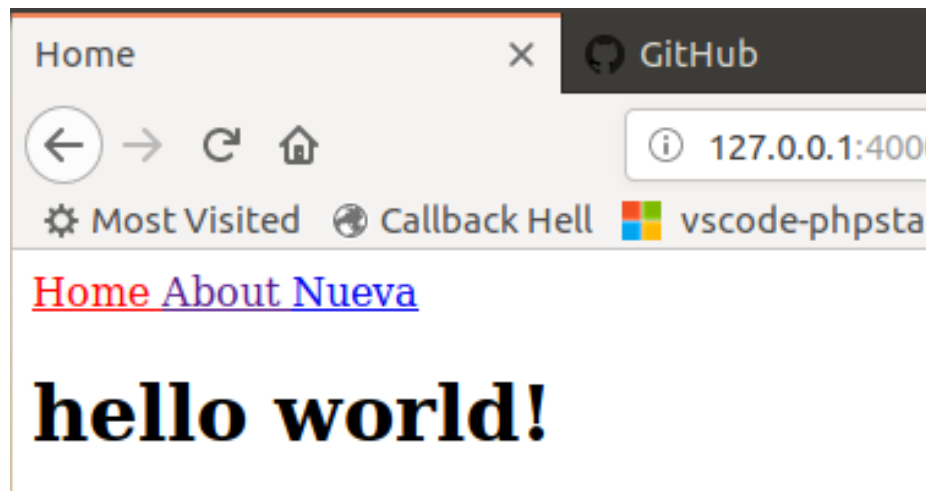


Figure 7: Menú con YAML

Assets

El uso de CSS, JS, imágenes y otros recursos es sencillo con Jekyll. Colócalos en la carpeta de tu sitio y se copiarán al sitio creado.

Los sitios de Jekyll a menudo usan esta estructura para mantener los activos organizados

```
.
  assets
|   css
|   images
|   js
...
```

SASS

Los estilos en línea utilizados en `_includes/navigation.html` no son las mejores prácticas, vamos a diseñar la página actual con una clase css.

```
{% raw %}
```

```
<nav>
  {% for item in site.data.navigation %}
    <a href="{ { item.link }}" {% if page.url == item.link %}class="current"{% endif %}>{ { item.title }}
  {% endfor %}
</nav>
```

```
{% endraw %}
```

Puedes usar un archivo CSS estándar para el estilo, vamos a ir un paso más allá usando Sass. Sass es una fantástica extensión de CSS incluida directamente en Jekyll.

Primero crea un archivo Sass en `/assets/css/styles.scss` con el siguiente contenido:

```
{% raw %}
```

```
---
---
```

```
@import "main";
```

```
{% endraw %}
```

El front matter vacío en la parte superior le dice a Jekyll que necesita procesar el archivo.

`@import "main"` le dice a Sass que busque un archivo llamado `main.scss` en el directorio sass (`_sass/` por defecto).

Así que creamos el archivo `_sass/main.scss` con el siguiente contenido:

```
.current {
  color: green;
}
```

Deberemos hacer referencia a la hoja de estilo en el diseño.

Abre `_layouts/default.html` y agrega la hoja de estilo al `<head>`:

```
{% raw %}

<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>{{ page.title }}</title>
    <link rel="stylesheet" href="/assets/css/styles.css">
  </head>
  <body>
    {% include navigation.html %}
    {{ content }}
  </body>
</html>

{% endraw %}
```

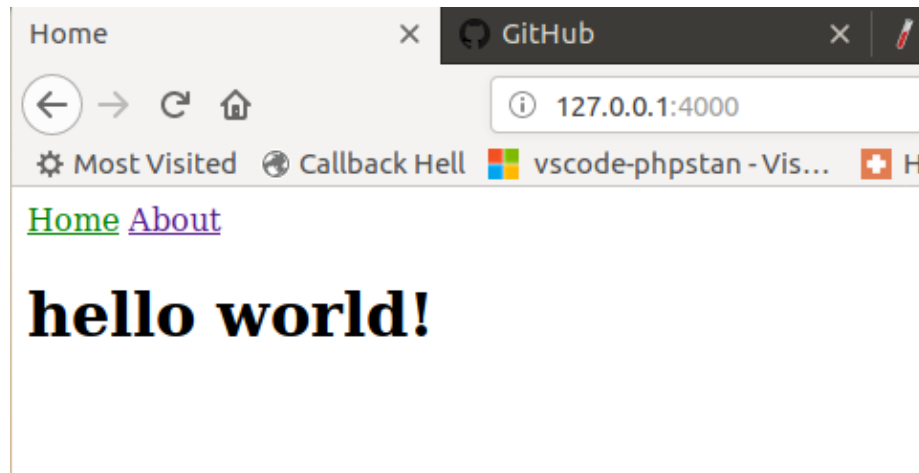


Figure 8: scss

Blogging

Quizás te preguntes cómo puedes tener un blog sin una base de datos. En el verdadero estilo Jekyll, los blogs solo funcionan con archivos de texto.

Posts

Las publicaciones de blog se almacenan en una carpeta llamada `_posts`. El nombre de archivo de las publicaciones tiene un formato especial: la fecha de publicación (**yyyy-mm-dd**), luego un título, seguido de una extensión.

Crea tu primera publicación en `_posts/2019-01-17-primera-entrada.md` con el siguiente contenido:

```
{% raw %}

---
layout: post
author: victor
---
# Esta es mi primera entrada de blog
Está en formato markdown, por lo se puede usar cualquier editor de markdown como typora o i

{% endraw %}

http://jmcglone.com/guides/github-pages/

https://help.github.com/articles/adding-a-jekyll-theme-to-your-github-pages-site-with-the-jekyll-theme-chooser/
```

Layout

El layout `post` no existe, así que vamos a crearlo con el siguiente contenido:

```
{% raw %}

---
layout: default
---
<h1>{{ page.title }}</h1>
<p>{{ page.date | date_to_string }} - {{ page.author }}</p>

{{ content }}

{% endraw %}
```

Este es un ejemplo de herencia de diseño. El layout `post` genera el título, la fecha, el autor y el cuerpo del contenido, que está incluido en el diseño predefinido (hereda de `default`)

También aplica el filtro `date_to_string`, que da formato a una fecha en un forma más agradable.

Listas de Posts

Actualmente no hay manera de navegar a la publicación del blog. Normalmente, un blog tiene una página que enumera todas las publicaciones, hagámoslo a continuación.

Jekyll pone las publicaciones disponibles en la variable `site.posts`.

Crea `blog.html` en tu raíz (`/blog.html`) con el siguiente contenido:

```
{% raw %}

---
layout: default
title: Blog
---
<h1>Latest Posts</h1>

<ul>
  {% for post in site.posts %}
    <li>
      <h2><a href="{{ post.url }}">{{ post.title }}</a></h2>
      <p>{{ post.excerpt }}</p>
    </li>
  {% endfor %}
</ul>

{% endraw %}
```

Hay algunas cosas a tener en cuenta con este código:

- Jekyll establece automáticamente `post.url` en la ruta de salida de la publicación.
- `post.title` se extrae del nombre de archivo de la publicación y se puede anular configurando el título en el front matter.
- `post.excerpt` es el primer párrafo de contenido por defecto

También necesitamos una forma de navegar a esta página a través de la navegación principal. Abre `_data/navigation.yml` y agrega una entrada para la página del blog:

```
- name: Home
  link: /
- name: About
  link: /about.html
- name: Blog
  link: /blog.html
```

Y este es el resultado:



Figure 9: Lista de blogs



Figure 10: Entrada de blog

Más posts

Crear una nueva entrada es tan sencillo como crear otro archivo en `_posts`

Por ejemplo, `_posts/2020-11-11-headless-cms.md`

```
{% raw %}
```

```
---
```

```
layout: post
```

```
author: victor
```

```
title: ¿Qué es un Headless CMS?
```

```
excerpt: En esta entrada se explica qué es un Headless CMS
```

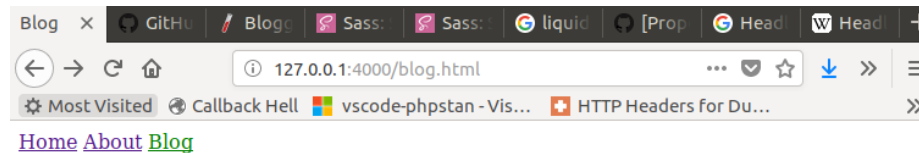
```
---
```

```
# Headless CMS
```

A headless content management system, or headless CMS, is a back-end only content management

The term "headless" comes from the concept of chopping the "head" (the front end, i.e. the v

```
{% endraw %}
```



Latest Posts

- [Primera Entrada](#)

Esta es mi primera entrada de blog

Está en formato markdown, por lo se puede usar cualquier editor de markdown como typora o instalar una extensión para Visual Studio Code.

- [¿Qué es un Headless CMS?](#)

En esta entrada se explica qué es un Headless CMS

Figure 11: Nueva entrada

Actualizar nuestro repo en GitHub Pages

Ya sólo nos queda subir los cambios a **GitHub** y ya tenemos nuestra página personal.

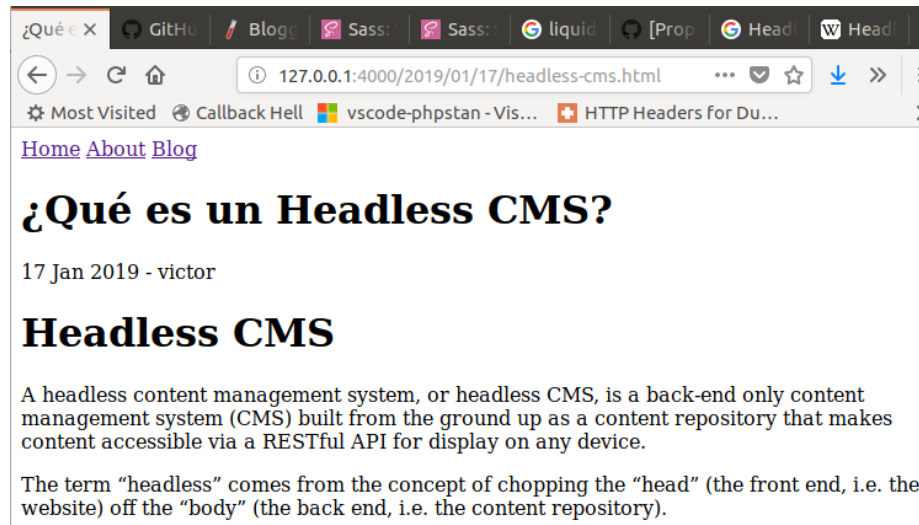


Figure 12: Otra entrada

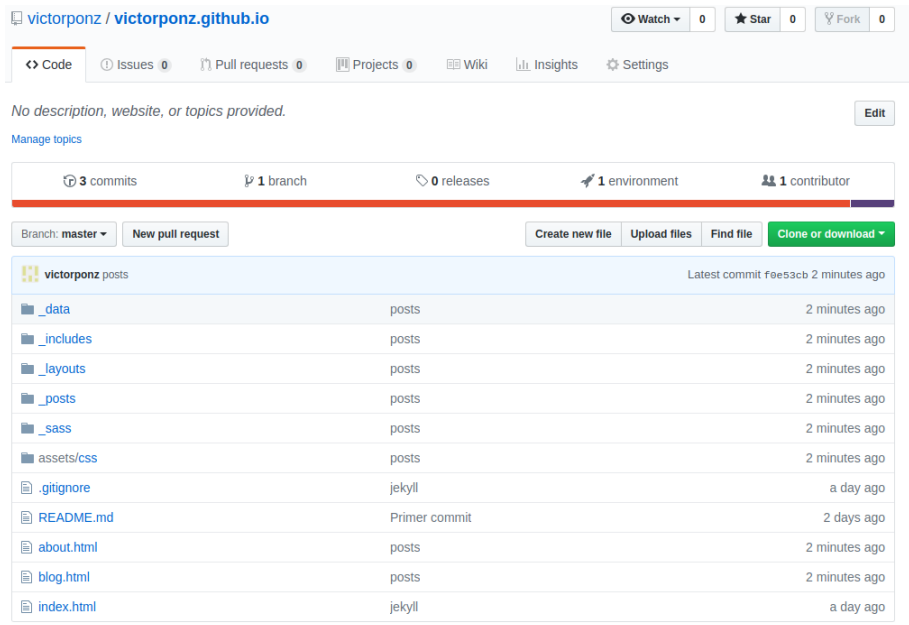


Figure 13: Repositorio actualizado

Una vez subimos los archivos, GitHub tiene que construir el sitio mediante Jekyll. Así que a veces no es automático.

Para ver el estado de vuestro sitio, id a **Settings**

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

✓ Your site is published at <https://victorponz.github.io/>

Source

Your GitHub Pages site is currently being built from the master branch. [Learn more.](#)

master branch ▾

Save

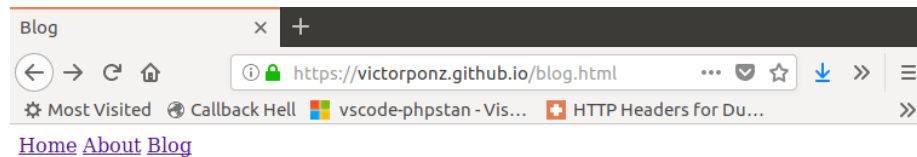
User pages must be built from the master branch.

Figure 14: GitHub Pages Settings

En este caso dice:

Your GitHub Pages site is currently being built from the master branch

A veces tarda bastante en actualizar, pero está en ello!



Latest Posts

• [Primera Entrada](#)

Esta es mi primera entrada de blog

Está en formato markdown, por lo se puede usar cualquier editor de markdown como typora o instalar una extensión para Visual Studio Code.

• [¿Qué es un Headless CMS?](#)

En esta entrada se explica qué es un Headless CMS

Figure 15: Actualizado

Uso de GitHub Pages sin Jekyll

No es necesario realizar nuestro sitio con **Jekyll**. Lo podemos realizar con cualquier **Headless CMS** e incluso hacerlo a la antigua usanza.

Otra opción posible es hacerlo con **Jekyll** pero que nuestro repositorio apunte al directorio `_site` en vez de a raíz. De esta forma al hacer un `git push` no nos hemos de esperar a que **GitHub** procese nuestro sitio.

Por ejemplo, supongamos que hemos hecho ya un `commit` de `_site` al repositorio:

```
victor@pedrablava:~/Documentos/jekyll/victorponz.github.io/_site$ git status
En la rama master
nothing to commit, working tree clean
```

Figure 16: Tree clean

Ahora modificamos `_layouts/post.html`:

```
{% raw %}

---
layout: default
---
<h1>V́ctor Ponz</h1>
<h1>{{ page.title }}</h1>
<p>{{ page.date | date_to_string }} - {{ page.author }}</p>

{% endraw %}
```

Y ahora el estado del repositorio es:

```
victor@pedrablava:~/Documentos/jekyll/victorponz.github.io/_site$ git status
En la rama master
Cambios no preparados para el commit:
  (use «git add <archivo>...» para actualizar lo que se confirmará)
  (use «git checkout -- <archivo>...» para descartar cambios en el directorio de trabajo)

modificado: 2019/01/17/headless-cms.html
modificado: 2019/01/17/primer-entrada.html
```

Figure 17: Modificado

Por lo que ya podría subir nuestros cambios a **GitHub Pages**

Ejemplos

En la siguiente dirección encontraréis más ejemplos de páginas personales.

<https://github.com/topics/personal-website>

Blog prediseñado para impacientes

Podemos crear un blog ya prediseñado haciendo uso de los siguientes comandos:

1. Creamos el blog con el nombre `myblog`

```
jeekyll new myblog
```

2. Cambiamos al directorio `./myblog`

```
cd myblog
```

3. Creamos el sitio y los publicamos en el servidor

```
bundle exec jeekyll serve
```

4. Navegamos a `127.0.0.1:4000`

Disqus

Vamos a integrar **Disqus** en nuestra página personal.

Disqus is a networked community platform used by hundreds of thousands of sites all over the web. With Disqus, your website gains a feature-rich comment system complete with social network integration, advanced administration and moderation options, and other extensive community functions. Most importantly, by utilizing Disqus, you are instantly plugging into our web-wide community network, connecting millions of global users to your small blog or large media hub.

Disqus works on just about any type of website or blog and can be installed either with a drop-in code snippet or by using one of the plugins available on our Install page. You can also customize and tweak Disqus for your website with extensive APIs and JavaScript hooks. Check out the Quick Start Guide or visit our homepage for more information and a demo of Disqus.

El primer paso es registrarse en **Disqus**.

Una vez os habéis registrado, seleccionad `I Want to install Disqus on my site`

Y completad el formulario con la url de vuestra página personal de **GitHub**

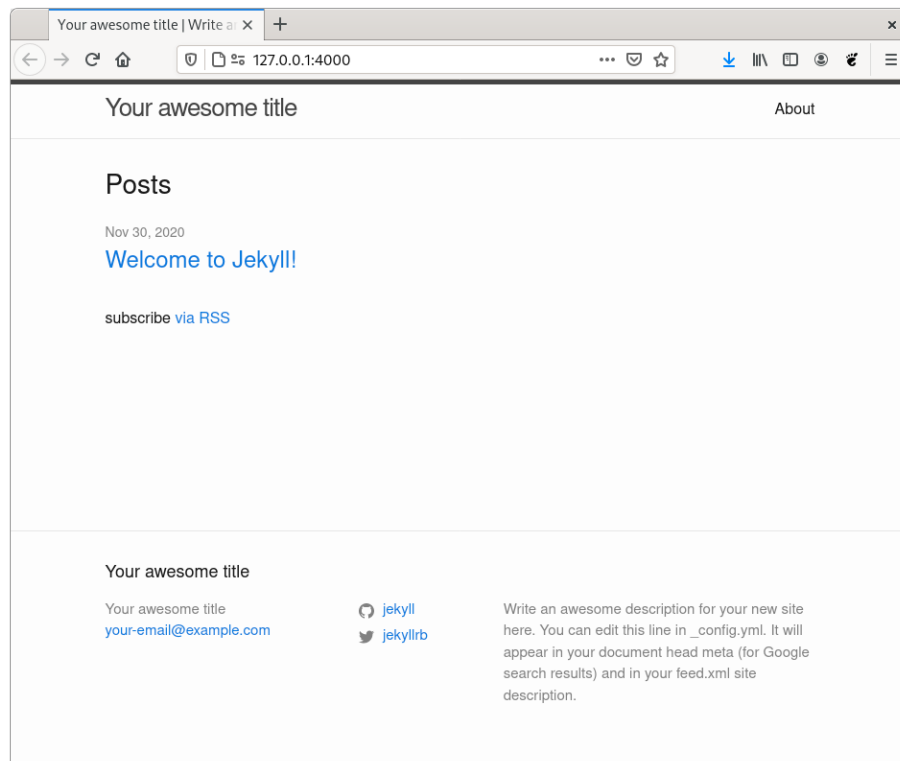


Figure 18: Blog prediseñado

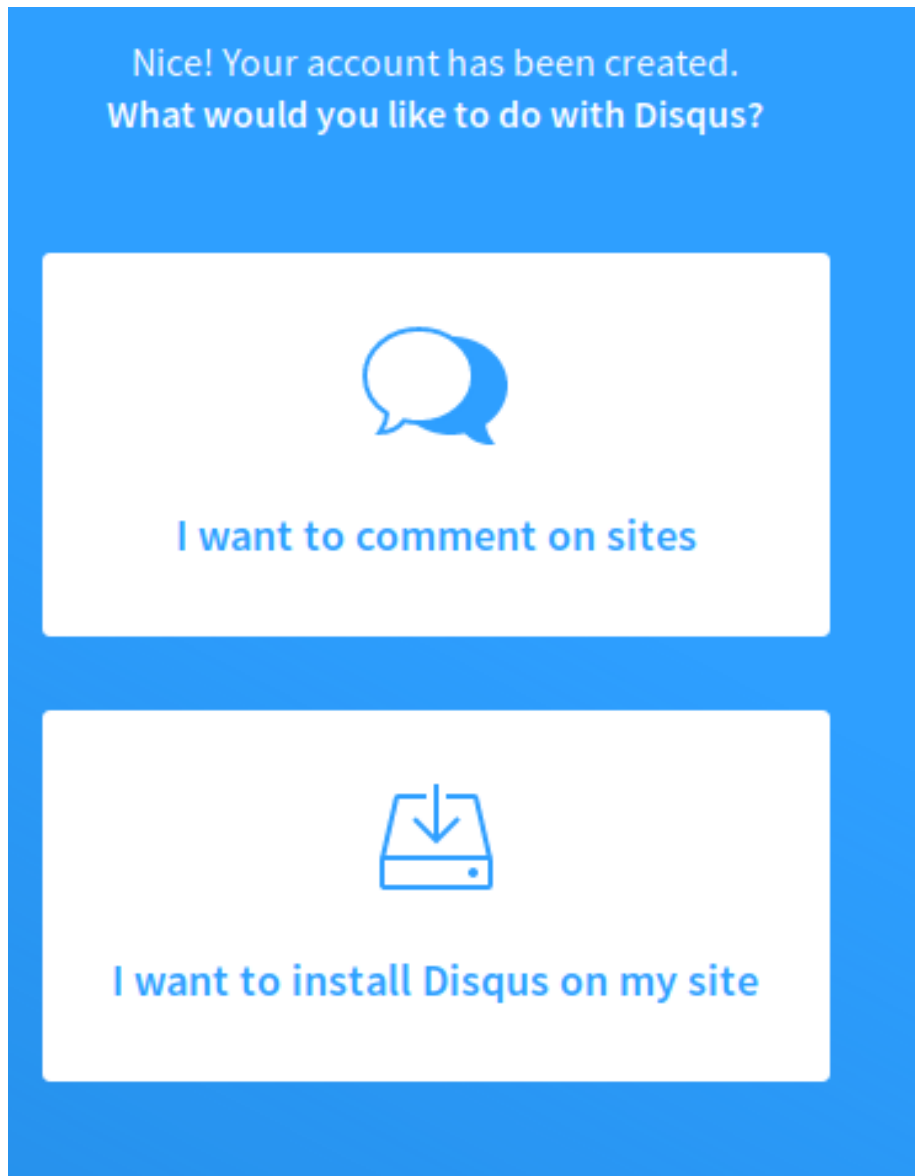



Figure 19: Disqus

Create a new site

All fields are required.

Site Owner

 victor ponz

To associate a different account as the site owner,
[login with a different account](#)

Website Name

Your unique disqus URL will be: <https-victorponz-github-io.disqus.com>
[Customize Your URL](#)

Category

Create Site

Figure 20: New Site

The image displays four pricing plans for Disqus, arranged in a grid. The 'Basic' plan is free with ads supported, allowing any number of daily pageviews. The 'Plus' plan costs \$19 per month (discounted from \$99) and is limited to under 50,000 daily pageviews. The 'Pro' plan costs \$89 per month (discounted from \$99) and is limited to under 150,000 daily pageviews; it is marked as 'POPULAR'. A 'Free' plan is also shown, intended for small, non-commercial sites without ads. Each plan lists its features and includes a 'Subscribe Now' or 'Start Trial' button. A note for all plans states 'No credit card required'.

Plan	Price	Pageviews	Key Features	Action
Basic	Free, Ads Supported	Any number	Core Disqus features, Comments Plug-in, Advanced Spam Filters, Moderation Tools, Basic Analytics, Configurable Ads, Reactions	Subscribe Now
Plus	\$19 \$9 per month	Under 50,000	Everything in Basic, Direct Support, Ads Optional	Start Trial (30 days)
Pro	\$99 \$89 per month	Under 150,000	Everything in Plus, Priority Support, Advanced Analytics, Shadow Banning, Timeouts, Email Subscriptions	Start Trial (30 days)
Free	Free	For small, personal, non-commercial sites	Everything in Plus except Direct Email Support	Subscribe Now

Figure 21: Free Plan

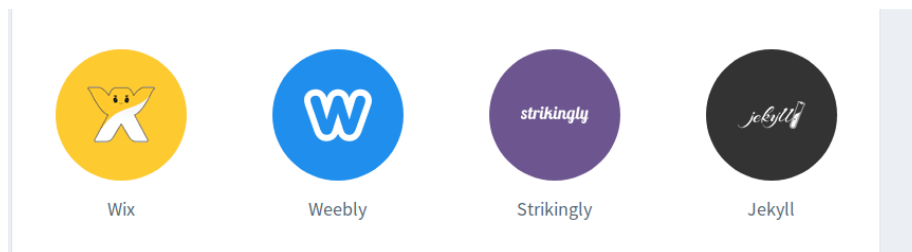


Figure 22: Jekyll CMS

Seleccionad la opción **Free!!**

Finalmente, elegid Jekyll

Ahora lo único que hay que hacer es modificar `post.html`, añadiendo el código que provee **Disqus**

```
<hr>
<div id="disqus_thread"></div>
<script>
var disqus_config = function () {
    this.page.url = 'https://username.github.io/{page.url}'; // Replace PAGE_URL with your page's url
};
(function() { // DON'T EDIT BELOW THIS LINE
var d = document, s = d.createElement('script');
s.src = 'https://https-username-github-io.disqus.com/embed.js';
s.setAttribute('data-timestamp', +new Date());
(d.head || d.body).appendChild(s);
})();
</script>
<noscript>Please enable JavaScript to view the <a href="https://disqus.com/?ref_noscript">comen
```

Reemplazando `username` por tu página personal de **GitHub** y con el código la url que te ha generado **Disqus**

Aquí os dejo el enlace para instalarlo en Jekyll, por si tenéis algún problema.

Y este es el resultado:

Tarea - Página con Jekyll

Crea tu página personal con jekyll Ten en cuenta que todas las prácticas de este módulo las crearás en tu página personal de GitHub



Figure 23: Disqus