PosaDev 2020

Documentation as Code

con Gitlab y Hugo

Enrique Cuevas

05 Diciembre 2020

¿Qué es Doc as Code?

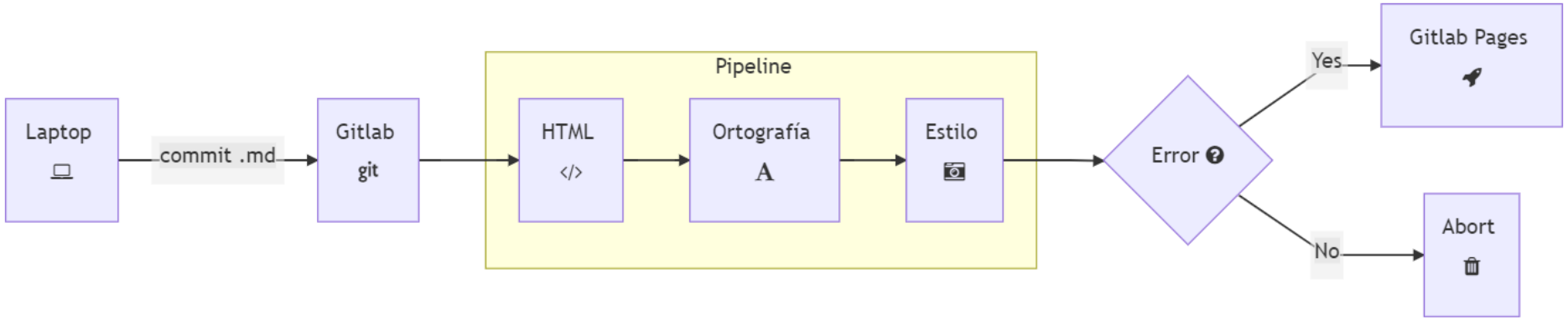
> Documentation as Code (`Docs as Code`) es un una técnica que nos dice que deberíamos escribir documentación de la misma manera en que escribimos código

- A través de `issues/merge` requests
- Usando control de versiones (Git)
- Archivos en texto plano (`Markdown`)
- Mediante colaboración y revisiones de código
- Implementando pruebas automatizadas (estilo, ortografía, links/imágenes)

Ventajas

- Automatizado
- Control de cambios
- Feedback inmediato
- Facilita la revisión de los cambios
- Validación de estilo y standards de escritura
- URL's e imágenes siempre funcionando.

Diagrama


Laptop
commit .md
Gitlab
git
Pipeline
HTML
</>
Ortografía
A
Estilo
Error
Yes
Gitlab Pages
No
Abort

Herramientas

- Hugo
- hugo-theme-learn
- Editor de Texto
- Gitlab

Demo

- ¿Qué vamos a ver?
- Estructura simple:

```
1. Empiece aquí
   * Instalación
      * Android
      * iOS
      * Linux
      * OSX
      * Windows
   * Configuración
2. Ejemplos
3. API
4. Hacks
```

Crear Proyecto

- Crear repo `asistente` usando **pages/hugo** templates
- Clonar repo en `vscode` (configurar ssh primero)
- Abrir `gitlab-ci.yml`
- Editar `config.toml`

```
baseurl = "https://rastangineer.gitlab.io/asistente/"
title = "Asistente"
```

- Revisar `Repo Settings -> Pages` y abrir url.

Agregar `hugo-learn-theme`

- Agregar hugo-learn-theme (búsqueda, soporte multi-idiomas, botones navegación, adjuntar archivos, diagramas mermaid)

```
git submodule add https://github.com/matcornic/hugo-theme-learn.git themes/hugo-theme-learn
```

- Modificar `config.toml`

```
theme = "hugo-theme-learn"
```

- ⚠ El pipeline fallará porque existen algunos archivo `markdown` del tema anterior que no son compatibles con el nuevo tema. Hay que borrarlos

```
Error: Error building site: "/builds/rastangineer/asistente/content/post/2017-03-20-photoswipe-gallery-sample.md:10:1": failed to extract shortcode: template for shortcode "gallery" not found
```

Arreglar el tema

- Borrar las carpetas:
  - `content/post/`
  - `content/page/`
  - `themes/beautifulhugo`
  - `themes/Lanyon`

Cambiar logo

- Crear archivo `layouts/partials/logo.html` y agregar

```
<a href="/">
   <img src="/logo.png" alt="logo.png">
</a>
```

- Copiar `logo.png` en la carpeta `static/`
- Cambiar color del tema:

```
[Params]
   # Change default color scheme with a variant one. Can be "red", "blue", "green".
   themeVariant = "green"
```

Editar página de inicio `content/_index.md`

```
# Hey Rasta

## Tu nuevo RastAsistente

* Realmente _inteligente_
* Privado. No compartimos tu información **con nadie**.
* Seguro
* Multiplataforma:
   * Android
   * iOS
   * Escritorio (Linux, Windows, Mac)
* Compatible con Siri y Google Assistant
* [Open Source](https://gitlab.com/rastangineerr/) and Free
* API
* SDK
* Hecho con amor

Dile hola a tu nuevo asistente, el **Rastasistente** 💗 ♡ ♡

### Acerca de esta documentación

Este portal fue generado con GitLab Pages / [Hugo](https://gohugo.io/) and [hugo-theme-learn]
(https://github.com/matcornic/hugo-theme-learn).
```

hugo-learn-theme

- Mejorar el `look and feel` con Awesome Font Icons
  - ⚠ Agregar el siguiente parámetro a `config.toml` para que los íconos se desplieguen correctamente:

```
[markup]
   defaultMarkdownHandler = "blackFriday"
```

- Notices
- Agregar Iconos y Notices (`content/_index.md`)

Capitulo 1. Empiece aquí

- Crear archivo `content/empiece-aqui/_index.md` y agregar:

```
+++
title = "Empiece aquí"
pre = "<b>1. </b>"
weight = 1
chapter = true
+++

## Hey Rasta!

### Descubra todo acerca del nuevo RastAsistente
```

Capitulo 2. Instalación

- Crear archivo `content/empiece-aqui/instalacion/_index.md`

```
+++
title = "Instalación"
weight = 2
chapter = false
+++

**RastAsistente** está disponible para todos los dispositivos y sistemas operativos.

* Android <i class="fab fa-android"></i>
* iOS <i class="fab fa-app-store-ios"></i>
* Linux <i class="fab fa-linux"></i>
* Windows <i class="fab fa-windows"></i>
* OSX <i class="fab fa-apple"></i>
```

Capítulo 2. Instalación -> Android

- Crear archivo `content/empiece-aqui/instalacion/android/_index.md`

```
+++
title = 'Android'
weight = 2
chapter = false
+++
# <i class="fab fa-android"></i>
![googleplay](https://play.google.com/intl/en_us/badges/images/generic/es_badge_web_generic.png?
width=20pc)
![f-droid](https://f-droid.org/badge/get-it-on.png?width=20pc)
```

Resto de capitulos

Ejemplos

API

Hacks

Reorganizar Pipeline

```yaml
image: registry.gitlab.com/pages/hugo:latest
variables:
  GIT_SUBMODULE_STRATEGY: recursive

stages:
  - build
  - deploy

generarHTML:
  stage: build
  script:
    - hugo -d test
  artifacts:
    paths:
    - test

pages:
  stage: deploy
  script:
    - hugo
  artifacts:
    paths:
    - public
  only:
  - master
```

Pipeline: Validar estilo y consistencia

- Se encarga de revisar el **estilo** de redacción, **standards** y **consistencia** de los archivos `Markdown`
- Agregar el siguiente código a `.gitlab-ci.yml`

```yaml
markdownlint:
  stage: test
  variables:
    markdownlint_cli_version: "0.22.0"
  image: peterdavehello/markdownlint:$markdownlint_cli_version
  script:
    - markdownlint content/
```

Pipeline: Validar estilo y consistencia (Fix)

- Crear archivo `.markdownlint.yml` y habilitar/deshabilitar reglas

```
{
"default": true,
"MD013": false, # Line length [Expected: 80; Actual: 90]
"MD033": false, # Inline HTML [Element: i]
"MD025": false, # Multiple top level headings in the same document
"MD026": false, # Trailing punctuation in heading [Punctuation: '!']
"MD014": false # Dollar signs used before commands without showing output [Context: "$ sudo apt-get
update"]
}
```

Pipeline: Revisión Ortográfica

- Se encarga de buscar errores ortográficos en los archivos Markdown
- Agregar el siguiente código a `.gitlab-ci.yml`

```
spellcheck:
  stage: test
  image: tmaier/hunspell
  script:
    - wget http://download.services.openoffice.org/contrib/dictionaries/es_MX.zip
    - unzip es_MX.zip
    - cp *.aff *.dic /usr/share/hunspell
    - export HUNSPELL_FINDINGS=`hunspell -d es_MX,en_US -p .spelling -u3 -H test/**/*.html | sort |
uniq`
    - echo "$HUNSPELL_FINDINGS"
    - test "$HUNSPELL_FINDINGS" == ""
```

Pipeline: Revisión Ortográfica (Fix)

- Crear `.spelling` y personalizar diccionario

```
RastAsisterasta
RastAsistente
iOS
OSX
png
Grav
API
AYBABTU
```

Pipeline: Validar HTML

- Entre otras cosas, valida:
  - Imágenes (alt tags, internal broken refs, external imgs,)
  - Links (external/internal broken links, internal hash references,)
  - Scripts (internal refs, external loading)
  - Favicon
- Agregar el siguiente código a `.gitlab-ci.yml`

```yaml
htmlproofer:
  image: 18fgsa/html-proofer:gitlab-ci
  script:
    - htmlproofer test --empty-alt-ignore --allow-hash-href
```

- ⚠ El pipeline fallará porque una URL no es válida. Corregir y continuar

Casos de uso

- Blogs (Rastangineer Blog)
- Documentación de Sotware/Productos/Servicios
- Tutoriales
- Libros/Cuentos (Cuentos Pachecos)
- Diarios
- Etc..
- CV online

Q & A

- PDFs?
- Soporte para comentarios? Discuss, Github issues