

# Documentación de aplicacións

<b>Documentación de aplicacións</b>	<b>1</b>
<b>VuePress</b>	<b>1</b>
Node.js	2
NPM	2
Yarn	2
<b>Proxecto VuePress</b>	<b>3</b>
<b>Estrutura de directorios</b>	<b>4</b>
<b>Funcionamento VuePress</b>	<b>6</b>
<b>Contido VuePress</b>	<b>8</b>
<b>Páxinas GitHub</b>	<b>12</b>
<b>Páxinas GitLab</b>	<b>14</b>
Creación dun repositorio VuePress en GitLab	15
Clonación do repositorio no equipo local	18
Funcionamento	19
<b>Referencias</b>	<b>21</b>

# Documentación de aplicacións

Hoxe en día é habitual acceder a internet para consultar a documentación de calquera software, API ou aplicación, pois cada vez é máis común que a documentación estea publicada nun sitio web. Estes sitios web poden crearse codificando manualmente as páxinas web coa linguaxe HTML e definindo os estilos con CSS ou utilizar unha alternativa que automatice a tarefa.

Dado que as páxinas de documentación son estáticas, é posible usar un **xerador de sitios estáticos** para crealas, simplificando o proceso da súa elaboración gracias a deseños e compoñentes predefinidos.

Un xerador de sitios estáticos (**SSG: Static Site Generator**) é un programa que xera un sitio web, utilizando arquivos de entrada de texto, como Markdown.

## VuePress

[VuePress](#) é un xerador de sitios estáticos creado por Evan You, creador de [Vue.js](#). Orixinalmente foi pensado para escribir a documentación dos propios proxectos Vue.

En realidade, un sitio VuePress é unha páxina web que utiliza Vue.js, Vue Router e webpack como base.

VuePress permite escribir o contido das páxinas web en ficheiros [markdown](#) e ofrece ferramentas para a creación automática do sitio web. Estas ferramentas realizan a conversión dos ficheiros markdown en páxinas HTML automaticamente.

Hoxe en día, cando se usa VuePress para xerar documentación, as páxinas xeradas terán un aspecto xeral parecido ao da páxina de [VuePress](#) (aínda que tamén hai dispoñibles outros temas).

Algunhas das características de VuePress son:

- Markdown optimizado para escribir documentación técnica.
- Posibilidade de utilizar Vue dentro de arquivos markdown.
- Sistema de temas baseado en Vue.
- Integración con Google Analytics.

VuePress inclúe un tema por defecto coas seguintes características:

- Deseño adaptable.
- Páxina de inicio opcional.
- Sistema de busca.
- Barra de navegación e barra lateral personalizable.

Para utilizar VuePress é necesario ter instalado [Node.js 10+](#) e [Yarn Classic](#) (opcional).

Aínda que Yarn é opcional, nestes apuntes vaise utilizar. Na [documentación oficial de VuePress](#) aparecen as dúas opcións de comandos a executar: **yarn** ou **npm**.

## Node.js

Node.js é unha contorna de execución de JavaScript. Inicialmente JavaScript só se podía executar dentro dun navegador. Node.js nace coa idea de poder executar JavaScript máis alá dun navegador, máis en concreto, nunha máquina como unha aplicación independente. Utilizando Node.js pode escribirse código do lado do servidor usando JavaScript, sendo posible interactuar con bases de datos e co sistema de arquivos, o que non estaba permitido dende o navegador.

Node.js utiliza o motor V8 de JavaScript, o mesmo que usa Chrome.

Para instalar Node.js pode [descargarse un instalador](#) para os diferentes sistemas operativos ou [instalalo a partir do xestor de paquetes](#). Por exemplo, para equipos baseados en [Debian e Ubuntu](#) habería que seguir estas instrucións (escoller a versión a instalar).

Para comprobar que Node.js está instalado, executar o seguinte comando na terminal:

```
node -v
```

## NPM

NPM (Node Package Manager) é o xestor de paquetes por defecto de Node.js e está escrito ao 100% con JavaScript.

Un xestor/administrador de paquetes é unha colección de ferramentas de software que permiten automatizar o proceso de instalación, actualización, configuración e eliminación de programas de forma consistente.

NPM é unha ferramenta de liña de comandos para a instalación de paquetes e a administración de versións e dependencias dos paquetes de Node.js. É dicir, permitirá instalar de forma fácil e sinxela paquetes de JavaScript.

NPM contén unha biblioteca de paquetes enorme que proporcionan múltiples utilidades para o desenvolvemento de aplicacións web.

Unha das grandes desvantaxes de NPM, e que se debe ter en consideración, é que todas as dependencias se instalan dentro dunha carpeta chamada `node_modules` de forma non determinista. Isto quere dicir que, segundo a orde na que se instalen as dependencias, a estrutura de `node_modules` pode variar. É dicir, dúas persoas desenvolvedoras poden ter unha estrutura diferente da carpeta `node_modules`.

Ao instalar Node, instálase automaticamente o seu xestor de paquetes e outras utilidades. Para comprobar que npm está instalado, executar o seguinte comando na terminal:

```
npm -v
```

## Yarn

En 2016 nace [Yarn](#) como un novo administrador de paquetes de código aberto debido ás limitacións de velocidade de NPM en proxectos moi grandes.

Yarn é un xestor de paquetes que permite usar e compartir código con outras persoas desenvolvedoras. É un xestor de paquetes de código rápido, seguro e confiable, que facilita o desenvolvemento do software.

Aínda que Yarn é un novo xestor de paquetes, é compatible co rexistro NPM, coa vantaxe de que opera máis rápido.

Un dos grandes problemas que ten NPM é a forma de descargar as dependencias (control de versións). Yarn resolve este problema facendo a instalación de maneira determinista, creando un arquivo **yarn.lock** que é onde almacena todas as versións exactas de cada un dos paquetes de forma concisa e ordenada, garantindo que cada instalación dá como resultado a mesma estrutura de arquivos na carpeta `node_modules` en todas as máquinas.

Outra vantaxe de usar Yarn é que primeiro busca nun directorio de caché global e se encontra o paquete, non será necesario descargalo.

Páxina de [axuda de instalación de Yarn](#).

Para comprobar que yarn está instalado, executar o seguinte comando na terminal:

```
yarn -v
```

## Proxecto VuePress

A forma máis rápida de crear un proxecto VuePress é usar o comando [create-vuepress-site generator](#) que creará a estrutura básica do proxecto de forma automática.

Para facelo, abrir un terminal e, no directorio desexado, executar o seguinte comando:

```
npx create-vuepress-site [optionalDirectoryName]
```

O comando anterior preguntará, de forma interactiva, polos detalles de configuración do proxecto: nome do proxecto, descrición, email, URL repositorio, ...

Unha vez contestadas as preguntas finaliza a execución do comando. Observar que se creou unha estrutura de arquivos e carpetas dentro do directorio **docs**.

Para ver o proxecto en funcionamento haberá que executar os seguintes comandos:

```
cd docs
yarn install
yarn dev
```

Solución de erros:

- [opensslErrorStack: \[ 'error:03000086:digital envelope routines::initialization error' \] - Stack Overflow](#)

```
export NODE_OPTIONS=--openssl-legacy-provider
```

## Estrutura de directorios

Durante a creación do proxecto de VuePress construíuse unha [estrutura de arquivos e carpetas](#) que seguen a convención de VuePress (**NOTA:** na documentación da web o directorio **docs** correspóndese co directorio **src** creado):

- **package.json:** almacena metadatos relevantes do proxecto. Contén información como a descrición do proxecto, autoría, versión, información da licenza, etc. Ademais, almacena información sobre as dependencias do proxecto cos nomes e versións de todos os paquetes instalados no mesmo. Esta información é utilizada polos xestores de paquetes (npm ou yarn).

O arquivo actualízase automaticamente cando se instalan paquetes ou dependencias no proxecto. A súa finalidade é manter o historial dos paquetes instalados e optimizar a forma en que se xeran as dependencias do proxecto e os contidos da carpeta **node\_modules**.

```
{
  "name": "documentacion",
  "version": "0.0.1",
  "description": "",
  "main": "index.js",
  "authors": {
    "name": "",
    "email": ""
  },
  "repository": "/documentacion",
  "scripts": {
    "dev": "vuepress dev src",
    "build": "vuepress build src"
  },
  "license": "MIT",
  "devDependencies": {
    "vuepress": "^1.5.3"
  }
}
```

Cando se instala un novo paquete no proxecto inclúese información do mesmo no ficheiro **package.json**. Os novos paquetes instalados almacénanse na carpeta **node\_modules**.

Na carpeta **node\_modules** instálanse os paquetes necesarios para o proxecto. Esta carpeta non debe subirse a Git nin a outro sistema de control de versións, porque ocuparía espazo innecesario. Calquera persoa que descargue o proxecto só necesita executar o comando **yarn install** para instalar todas as dependencias.

Para evitar subir a Git o directorio anterior, é necesario indicalo no ficheiro **.gitignore**. Pode utilizarse como exemplo un ficheiro [.gitignore para aplicacións que usen Node](#).

**devDependencies** inclúe o nome e versión dos paquetes de desenvolvemento. Este campo funciona a modo de guía ou índice de paquetes necesarios para o proxecto.

No ficheiro tamén hai uns scripts que inclúen as instrucións para executar o proxecto.

**OLLO:** os parámetros do ficheiro package.json deben ir separados por “,”.

- **src/.vuepress:** almacena configuración global, compoñentes, recursos estáticos, etc.
- **src/.vuepress/components:** compoñentes Vue.
- **src/.vuepress/styles:** ficheiros de estilos. Usan a [linguaxe Stylus](#) que é un CSS creado para nodejs.
- **src/.vuepress/config.js:** ficheiro de configuración do sitio, que debe exportar un obxecto JavaScript. Contén a definición de diferentes propiedades usadas para configurar o sitio.

Este ficheiro será diferente en función do tema a usar. Neste apartado verase a configuración do tema por defecto.

```
const { description } = require('.../package')

module.exports = {
  title: 'Vuepress Docs Boilerplate',
  description: description,
  // Extra tags to be injected to the page HTML `<head>`
  head: [
    ['meta', { name: 'theme-color', content: '#3eaf7c' }],
    ['meta', { name: 'apple-mobile-web-app-capable', content: 'yes' }],
    ['meta', { name: 'apple-mobile-web-app-status-bar-style', content: 'black' }],
  ],
  themeConfig: {
    repo: "",
    editLinks: false,
    docsDir: "",
    editLinkText: "",
    lastUpdated: false,
    nav: [
      { text: 'Guide', link: '/guide/' },
      { text: 'Config', link: '/config/' },
      { text: 'VuePress', link: 'https://v1.vuepress.vuejs.org' },
    ],
    sidebar: {
      '/guide/': [
        {
          title: 'Guide',
          collapsable: false,
          children: ['', 'using-vue'],
        },
      ],
    },
  },
  plugins: ['@vuepress/plugin-back-to-top', '@vuepress/plugin-medium-zoom'],
}
```

O título ([title](#)) é engadido en todas as páxinas e tamén na barra de navegación superior.

O atributo [description](#) será mostrado na etiqueta <meta> de HTML. Estará dispoñible nos ficheiro markdown utilizando a sintaxe `{{ $site.description }}`

Observar que cando o servidor se está executando, no navegador mostrarase a páxina con unha cabeceira formada polo título e unha caixa para buscar. VuePress inclúe automaticamente a funcionalidade para buscar nas cabeceiras das páxinas do sitio, é dicir crea un índice de busca co título das páxinas e as cabeceiras h2 e h3 de todas as páxinas.

[Base URL](#): se o proxecto se coloca nun directorio que non é a raíz do servidor, será necesario establecer o parámetro **base** no ficheiro `.vuepress/config.js`. Por exemplo, se se quere colocar o sitio web en `https://foo.github.io/bar/`, **base** debe establecerse a `"/bar/"` (debe empezar e acabar sempre con `"`).

Máis información na páxina de [referencia da configuración](#) e na [configuración do tema por defecto](#).

- **docs/.vuepress/public**: directorio cos recursos estáticos.

## Funcionamento VuePress

VuePress permite que as páxinas que constitúen o sitio estean escritas en linguaxe [markdown](#). Revisar a sintaxe markdown para entender como escribir documentos.

Os ficheiros Markdown son ficheiros de texto con extensión **md** e cun formato específico. Por exemplo, un ficheiro con unha cabeceira e un parágrafo que inclúa unha ligazón, escribiríase así:

```
# Start here
```

```
Quick note: novices may need to review the [glossary](/glossary.md) before continuing
```

Se tivéssemos que escribilo en HTML, habería que escribir o seguinte:

```
<h1>Start here</h1>
```

```
Quick note: novices may need to review the <a href="/glossary.html">glossary</a> before continuing.
```

O contido de VuePress estará estruturado en directorios con ficheiros Markdown (extensión md).

Os ficheiros Markdown son convertidos a ficheiros HTML co mesmo nome, mais coa extensión `.html`. **Excepción**: os ficheiros `README.md` ou `index.md` de cada subdirectorio serán convertidos automaticamente en **index.html**.

README.md debe empezar cunha cabeceira. Non ten por que ser H1 (#), mais pode ser H2 (##) ou H3 (###) para unha correcta xeración de índices e barras laterais.

Se o sitio web está composto por subdirectorios, estes deben conter un ficheiro README.md. Un subdirectorio é invisible a VuePress a menos que conteña un ficheiro README.md

Todos os ficheiros markdown son compilados en compoñentes Vue e procesados por webpack (empaquetador de módulos). Os recursos deben ser referenciados pola súa URL relativa:

```
![An image](./image.png) <!-- imaxe do directorio actual -->
![An image](/image.png) <!-- imaxe do directorio .vuepress/public -->
```

Así, a imaxe será procesada e copiada ao directorio apropiado á hora de crear o sitio web.

Cando sexa necesario proporcionar un recurso que non está referenciado en ningún ficheiro markdown, pode colocarse no directorio .vuepress/public e será copiado ao directorio raíz do proxecto xerado.

Polo tanto, unha vez creado o proxecto de VuePress, o fluxo normal de traballo é:

- Crear ficheiros markdown con extensión **.md**. A páxina inicial sempre será README.md.
- Executar o comando **yarn dev**. Este comando fai que se rexenere o sitio web en base aos ficheiros fonte e lanza un servidor que estará escoitando en `http://localhost:8080/`
- Acceder dende un navegador a <http://localhost:8080/>
- Cambiar os ficheiros markdown ou crear novos. VuePress ve os cambios e rexenera automatizadamente o sitio web.



# Contido VuePress

A continuación móstranse diferentes exemplos de contido que se pode agregar ás páxinas creadas con VuePress. Debe consultarse a documentación para máis información:

- [Páxina principal](#). O tema por defecto proporciona unha páxina principal con un layout predefinido. Para usalo hai que especificar **home: true** xunto con outros metadatos no ficheiro raíz **index.md** creado coa sintaxe de [YAML frontmatter](#). Exemplo:

```
---
home: true
heroImage: https://v1.vuepress.vuejs.org/hero.png
tagline:
actionText: Quick Start →
actionLink: /guide/
features:
- title: Feature 1 Title
  details: Feature 1 Description
- title: Feature 2 Title
  details: Feature 2 Description
- title: Feature 3 Title
  details: Feature 3 Description
footer: Made by with ❤️
---
```

O tema por defecto buscará a “heroImage” no directorio **.vuepress/public**. Para personalizar a imaxe a usar, debe colocarse neste directorio. [Máis información](#).

O tema por defecto inclúe un botón grande central que actúa como ligazón. Esta ligazón denomínase “action link” e está composta por **actionText** que é a etiqueta e **actionLink** que é a ruta da ligazón.

**features** utilízanse para destacar características da páxina xunto cunha pequena descrición. Observar que cando se inclúen na páxina web son “responsive”.

- [Ligazóns](#). Na creación da páxina web os ficheiros md son transformados a ficheiros html, polo que as ligazóns son adaptadas adecuadamente. Exemplos:

```
[Home](/) <!-- Crea unha ligazón ao README.md raíz do proxecto -->
[foo](/foo/) <!-- Crea unha ligazón a index.html of directory foo -->
[Google](http://www.google.com)
```

Unha ligazón a unha páxina externa créase automaticamente cos atributos **target="\_blank" rel="noopener noreferrer"**

As imaxes tamén se insiren como unha ligazón:

**![An image](./images/clockFace.png)**

Tendo en conta que **images** é un subdirectorio do directorio actual.

- [Barra de navegación](#): a barra de navegación pode conter o título da páxina, unha caixa de busca, ligazóns, etc.

[nav](#) define as ligazóns que aparecerán na barra de navegación superior.

As ligazóns tamén pode configurarse como menús dropdwon. Ver exemplo de [ligazóns na barra de navegación](#):

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    nav: [
      { text: "Guide", link: "/guide/" },
      { text: "Config", link: "/config/" },
      { text: "VuePress", link: "https://v1.vuepress.vuejs.org" },
    ]
  }
}
```

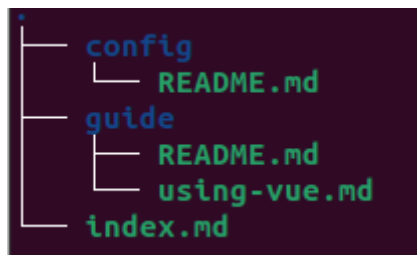


- [Barra lateral](#)

Pode establecerse unha barra lateral común para todas as páxinas do sitio:

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: {
      '/': ['/', '/guide/', '/config/'],
    },
  }
}
```

Tamén é posible [configurar barras laterais personalizadas para cada sección de contido](#). Para isto, primeiramente hai que organizar as páxinas en directorios para cada sección e despois definir a barra lateral para cada sección. Así, dada a seguinte estrutura de directorios:



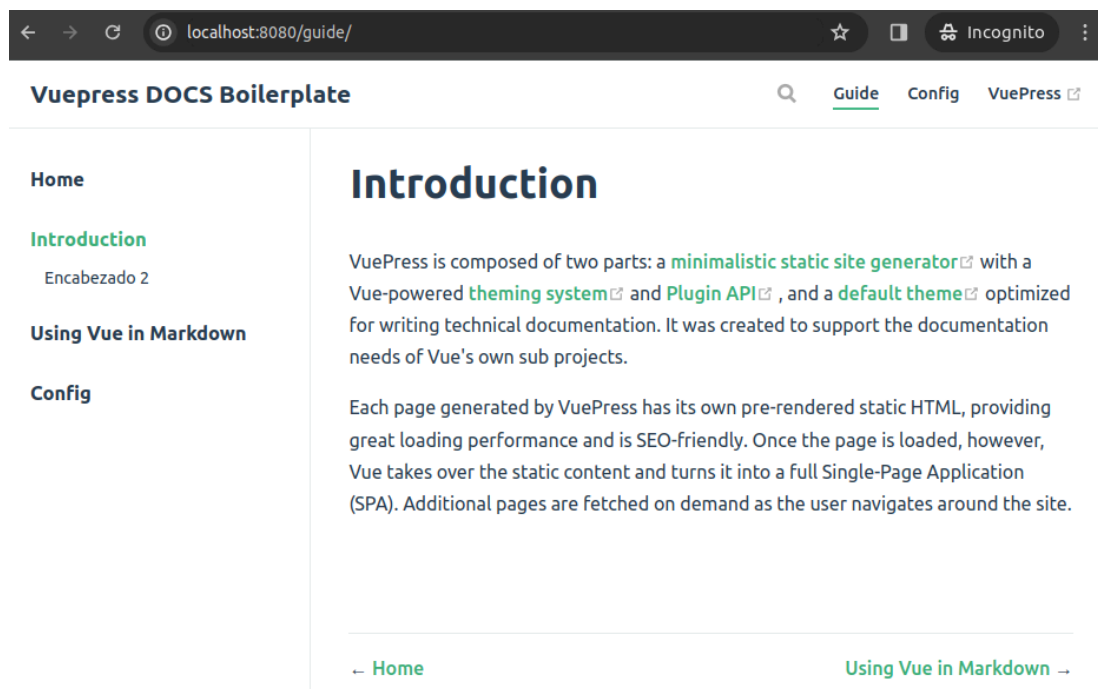
Poden definirse as barras laterais para cada sección:

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: {
      '/config/': [''],
      '/guide/': ['/', '', 'using-vue', '/config/'],
      '/': ['/', '/guide/', '/config/'],
    },
  },
}
```

**NOTA:** as páxinas enlazadas teñen que existir.

Observar que a última sección que se define é a '/'. Se se colocase de primeira, VuePress interpretaría a como a barra lateral definida para todas as seccións.

Observar que no caso anterior aparecen na parte de abaixo da páxina unhas ligazóns para navegar entre as páxinas da barra lateral: [ligazón anterior/seguinte](#).



Tamén pode ser útil e cómodo crear as barras laterais de forma automática que incluírán as cabeceiras da páxina actual. Para conseguilo simplemente hai que engadir ao principio de cada páxina o código YAML seguinte:

```
---
sidebar: auto
---
```

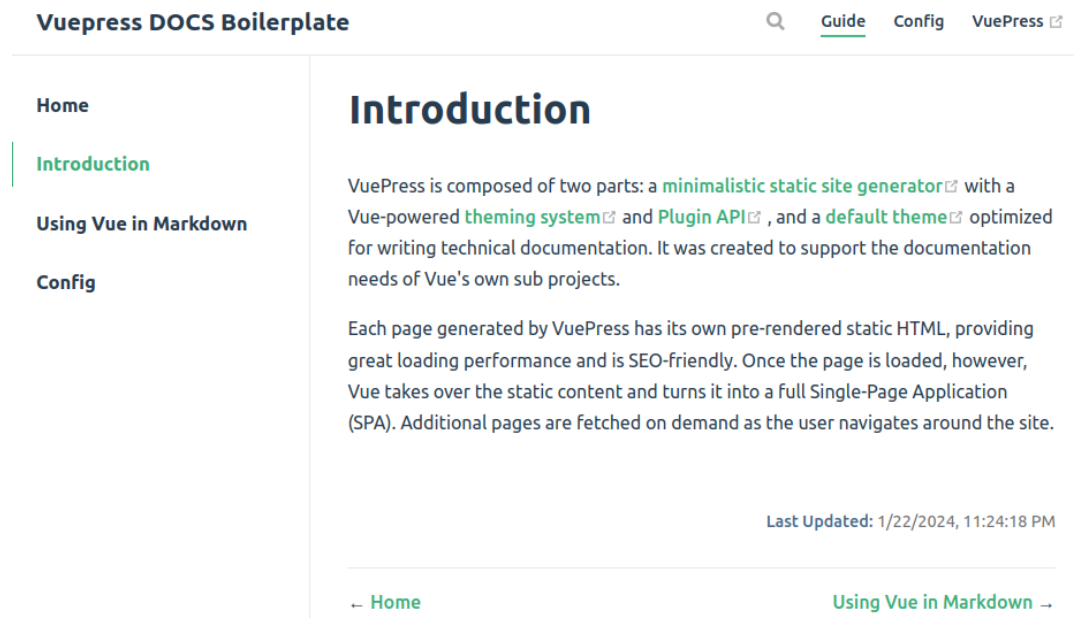
- [Táboas](#)
- [Incluír emojis](#). Consultar a [lista de emojis](#).

- [Contadores personalizados.](#)
- [Bloques de código coa sintaxe resaltada.](#)
- [Resaltar unha liña nun bloque de código.](#)
- Engadir unha [marca para indicar a última vez que foi modificada unha páxina.](#)

Engadir no ficheiro de configuración de vuepress a seguinte liña.

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    lastUpdated: 'Last Updated', // string | boolean
  }
}
```

Esta marca está baseada no timestamp de cada ficheiro de cando foi a última vez que se fixo un commit con git. Dado que está baseado en git, só se pode usar nun repositorio git.



Aínda que o mellor para aprender é practicar engadindo compoñentes e complementos que hai na [documentación](#).

### Exercicio:

1. Utilizando VuePress, crea un sitio web que inclúa a documentación necesaria para explicar o proceso de creación dun compoñente visual.

## Páxinas GitHub

[GitHub Pages](#) ofrece a posibilidade de transformar un repositorio de GitHub nun sitio web, sen necesidade de configurar unha base de datos nin un servidor.

GitHub ofrece unha URL personalizada (<https://your-account.github.io>) como acceso ao sitio web persoal.

Neste apartado detállanse os pasos a realizar para publicar o proxecto de VuePress nun repositorio de GitHub.

Realizar as seguintes operacións:

- Crear un proxecto VuePress.
- Cambiar o nome do directorio **src** por **docs**.
- Instalar os paquetes necesarios

```
npx create-vuepress-site [optionalDirectoryName]
cd docs
mv src docs
yarn install
```

```
/media/MV-Linux/repositorios/gitHubPages$ ls
docs
/media/MV-Linux/repositorios/gitHubPages$ cd docs/
/media/MV-Linux/repositorios/gitHubPages/docs$ ls
package.json  src
/media/MV-Linux/repositorios/gitHubPages/docs$ mv src/ docs
/media/MV-Linux/repositorios/gitHubPages/docs$ yarn install
```

**NOTA:** dado que o proxecto VuePress se subirá a un repositorio de GitHub, non se debe colocar dentro doutro repositorio de Git xa existente.

Modificar o ficheiro **package.json** coas seguintes instrucións:

```
{
  ...
  "scripts": {
    "dev": "vuepress dev docs",
    "build": "vuepress build docs",
    "docs:build": "vuepress build docs"
  },
  ...
}
```


A continuación, crearase un repositorio persoal que levará como nome a URL personalizada de acceso ao sitio web (**your-account.github.io**). O nome ten que coincidir coa conta de usuario exactamente:

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner \*

 cristina-paris ▾

Repository name \*

/ cristina-paris.github.io ✓

Great repository names are short and memorable. Need inspiration? How about **friendly-enigma**?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Crear o ficheiro **deploy.sh** co seguinte contido e **actualizar** a liña resaltada en negro co login apropiado de GitHub:

```
#!/usr/bin/env sh

# abort on errors
set -e

# build
npm run docs:build

# navigate into the build output directory
cd docs/.vuepress/dist

# if you are deploying to a custom domain
# echo 'www.example.com' > CNAME

git init
git add -A
git commit -m 'deploy'

# if you are deploying to https://<USERNAME>.github.io
# git push -f git@github.com:<USERNAME>:<USERNAME>.github.io.git master
git remote add origin git@github.com:<USERNAME>:<USERNAME>.github.io.git
git branch -M main
git push -u origin main

# if you are deploying to https://<USERNAME>.github.io/<REPO>
# git push -f git@github.com:<USERNAME>:<REPO>.git master:gh-pages

cd -
```

A continuación hai que executar o script `deploy.sh`. [Por problemas de compatibilidade en versións actuais de Node](#), hai que executar o script coa seguinte configuración de Node:

```
NODE_OPTIONS=--openssl-legacy-provider sh deploy.sh
```

O script anterior compila o proxecto e súbeo a GitHub, polo que unha vez remate o seu despregamente en GitHub, xa estará dispoñible o sitio web:



## Páxinas GitLab

Usando as [Páxinas de Gitlab](#), pódense publicar sitios web estáticos directamente dende un repositorio de GitLab.

Para publicar un sitio web usando as Páxinas de GitLab, pode usarse calquera xerador de sitios web. No noso caso usaremos **VuePress**.

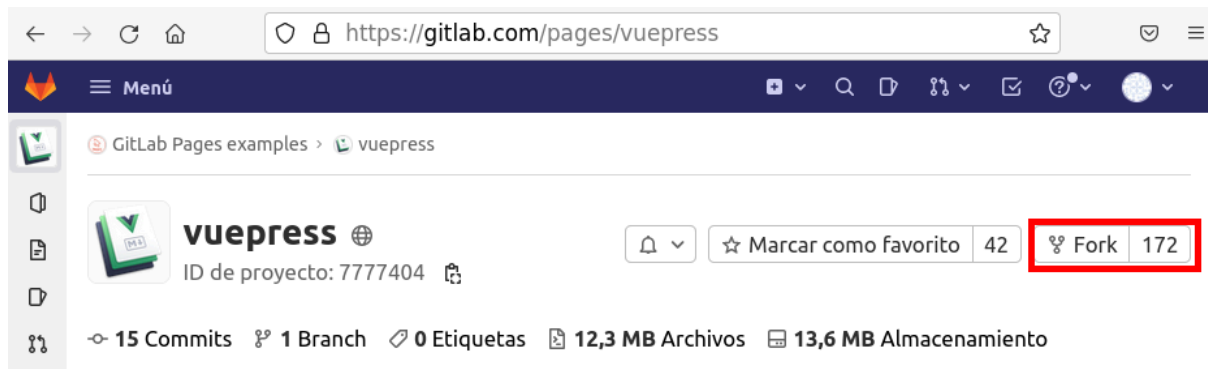
A creación do sitio web pode facerse de varias formas. Pódese programar todo a man partindo de cero ou usar como base un proxecto exemplo, que será o que imos facer. É dicir, clonaremos un repositorio exemplo e faremos as modificacións adaptadas ao noso caso.

GitLab Pages proporciona [exemplos de proxectos para os xeradores de sitios estáticos máis populares](#).

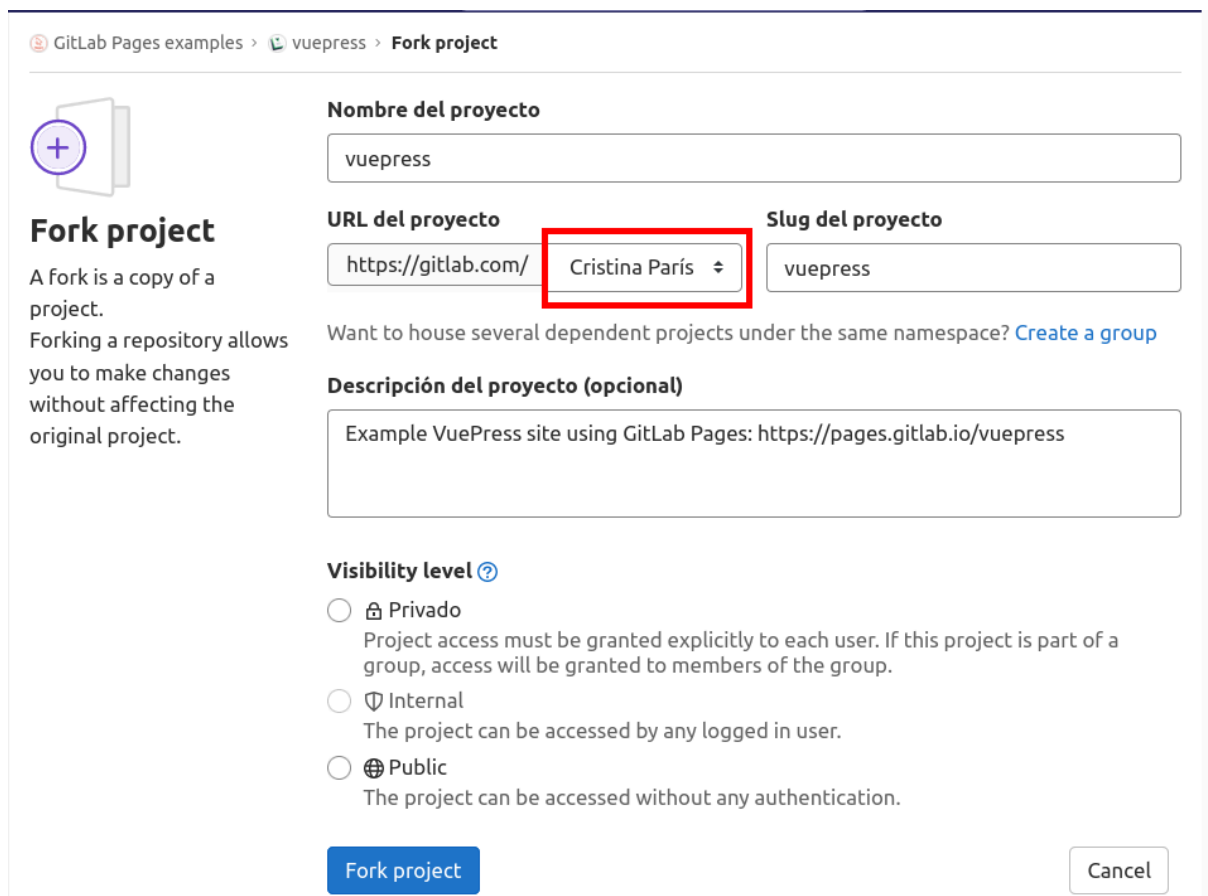
## Creación dun repositorio VuePress en GitLab

Neste apartado vaise mostrar como crear en GitLab un repositorio de VuePress.

Para iso imos partir do [repositorio exemplo de Vuepress](#) ao que imos facerlle un fork.



Seleccionamos como namespace o nome da nosa conta de GitLab e configuramos o repositorio como privado.





Unha vez creado o repositorio na nosa conta de GitLab e dado que non imos contribuír ao proxecto orixinal, imos eliminar a relación de fork. Para iso imos a **Configuración -> Xeral -> Avanzado ->**

### Eliminar la relación del fork

This will remove the fork relationship between this project and [GitLab Pages examples / vuepress](#).

Once removed, the fork relationship cannot be restored. This project will no longer be able to receive or send merge requests to the source project or other forks. [Obtener mas información](#).

Eliminar la relación del fork

Para realizar esta operación pídesse a confirmación escribindo a mensaxe indicada:

#### Se requiere confirmación



ⓘ You are going to remove the fork relationship from 2021-2022-2DAM / vuepress. Are you ABSOLUTELY sure?

Esta acción puede provocar la pérdida de datos. Para prevenir acciones accidentales, le pedimos que confirme su intención.

Por favor, escriba `vuepress` para continuar o cierre esta ventana modal para cancelar.

Se requiere confirmación

Confirmar

A continuación temos que construír o noso sitio web utilizando a ferramenta GitLab CI/CD (*Continuous Integration / Continuous Delivery / Continuous Deployment*). Para iso accedemos ao menú **CI/CD -> Pipelines** e executamos “Run pipeline”.

2021-2022-2DAM > vuepress > Pipelines

Todos 0 Finalizado Ramas Etiquetas

Clear runner caches

CI lint

Run pipeline

Filter pipelines



Show Pipeline ID ▾

Para executar o pipeline deixamos as opcións por defecto da seguinte imaxe:

2021-2022-2DAM > vuepress > Pipelines

---

### Run pipeline

---

Run for branch name or tag

master

---

Variables

Variable  Input variable key  Input variable value

Specify variable values to be used in this run. The values specified in [CI/CD settings](#) will be used by default.

[Run pipeline](#) [Cancelar](#)

Para ver o que sucede, debemos pulsar na páxina iniciada, esperar a que remate de construír o sitio web e comprobar cando o traballo finalizou con éxito.

```

54 Uploading artifacts for successful job
55 Uploading artifacts...
56 public: found 12 matching files and directories
57 Uploading artifacts as "archive" to coordinator... 201 Created id=2003497906 responseStatus=201 Created token=tgfBwPM6
59 Cleaning up project directory and file based variables
61 Job succeeded
  
```

Unha vez termine de executarse o pipeline temos o sitio web accesible dende as páxinas de GitLab. Para mirar a ruta pódese acceder a: **Configuración -> páxinas**.

Cristina París > vuepress > Páxinas

---

Q Search settings

---

### Páxinas [Nuevo dominio](#)

With GitLab Pages you can host your static website directly from your GitLab repository. [Learn more.](#)

☒ **Forzar HTTPS (requiere certificados válidos)**

When enabled, all attempts to visit your website through HTTP are automatically redirected to HTTPS using a response with status code 301. Requires a valid certificate for all domains. [Learn more.](#)

[Guardar los cambios](#)

---

Páxinas de acceso

---

Sus páginas se sirven desde:

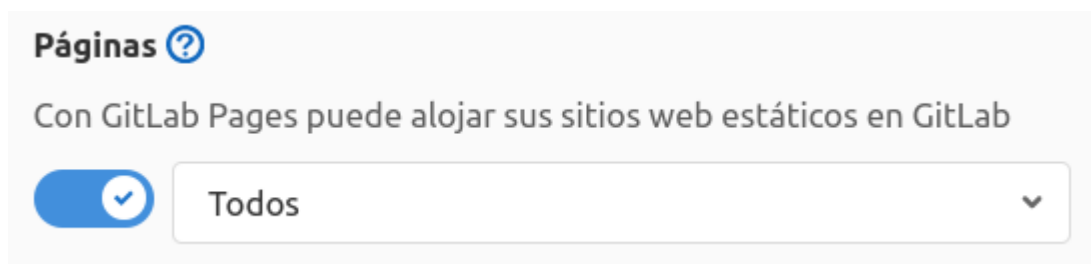
<https://cparis-fp.gitlab.io/vuepress>

No meu caso está na ruta <https://cparis-fp.gitlab.io/vuepress>



Despois, en cada cambio que se produza no repositorio, GitLab executará un novo pipeline e publicará automaticamente o sitio web actualizado.

Ademais, GitLab permite configurar o proxecto para decidir quen pode ver as súas páxinas. Isto faise na páxina do proxecto -> **Configuración** -> **Xeral** -> **“Visibilidade, características do proxecto, permisos”**. Escoller que todos poidan ver as **páxinas**.



## Clonación do repositorio no equipo local

Unha vez temos o repositorio en GitLab imos clonalo no equipo local para traballar nel utilizando o comando **git clone**.

A continuación imos construír o sitio en local. Para iso necesitamos ter yarn instalado.

Unha vez comprobado que está yarn instalado, para instalar as dependencias do proxecto hai que executar o comando, dentro do directorio do proxecto:

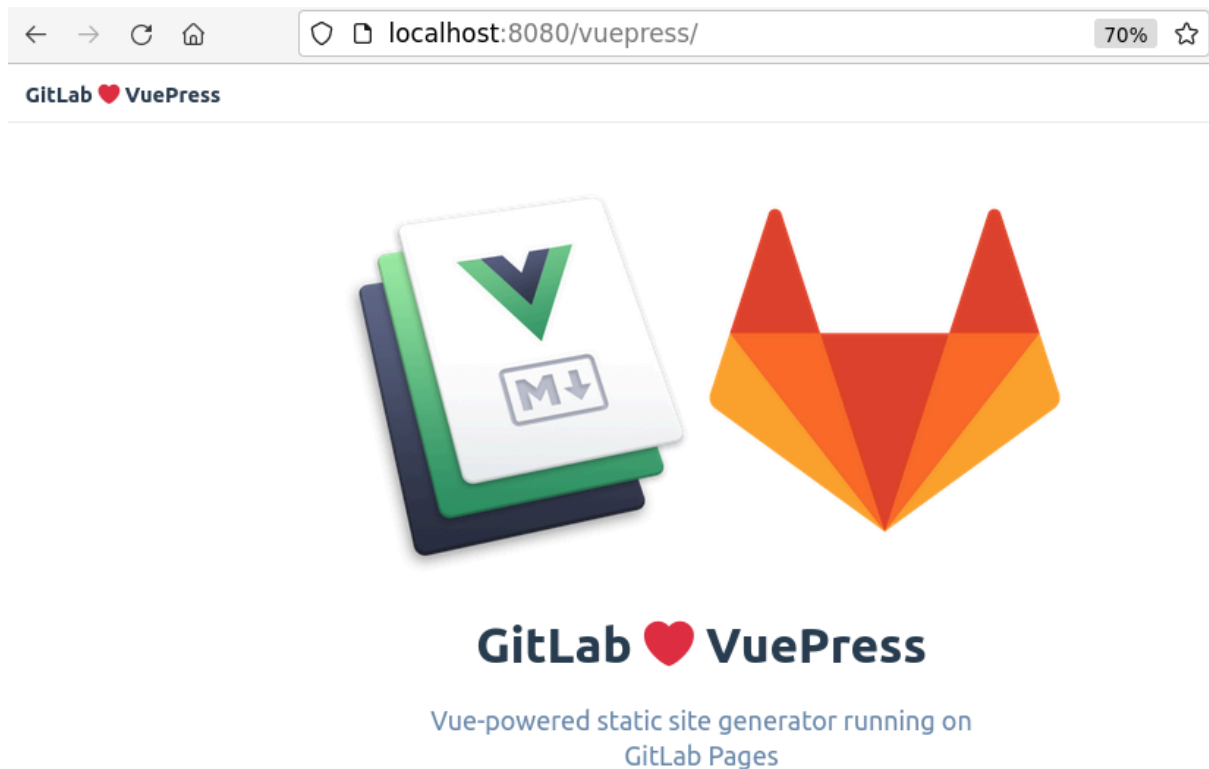
```
ruta/vuepress$ yarn install
```

E para construír o sitio web en local e lanzar o servidor web, executamos o seguinte comando:

```
ruta/vuepress$ yarn start
```

As instrucións que se executan con estes comandos están definidas no ficheiro **package.json**.

Unha vez o comando remata de executarse e o proxecto foi correctamente compilado, indícanos que se pode acceder ao sitio web a través da ruta <http://localhost:8080/vuepress/>



## Funcionamento

Para despregar o sitio web, GitLab usa a ferramenta **CI/CD** para construír as páxinas web e publicalas no servidor de páxinas de GitLab. Isto permite construír, testar e despregar o código automaticamente cada vez que se fai unha modificación no repositorio.

**CI/CD** fai referencia a:

- **Continuous Integration** significa que cada vez que se realice unha operación de push ao repositorio, executaranse unha serie de scripts para construír e testar a aplicación automaticamente.
- **Continuous Delivery** significa que a aplicación será despregada manualmente despois do paso anterior.
- **Continuous Deployment** é similar ao paso anterior coa diferenza de que neste caso o despregue se realiza automaticamente.

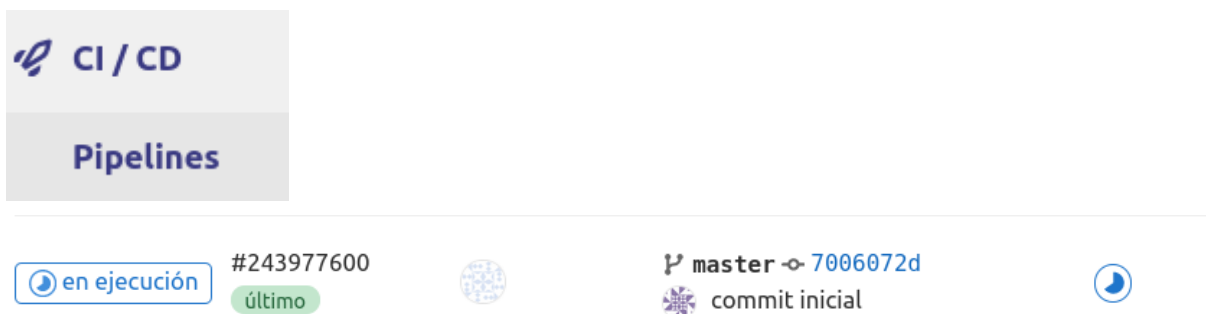
A secuencia de scripts que se executan para conseguir esta tarefa están almacenadas no ficheiro **.gitlab-ci.yml**, situado na raíz do repositorio:

```
image: node:9.11.1

pages:
  cache:
    paths:
      - node_modules/
  script:
    - yarn install
    - yarn run build
  artifacts:
    paths:
      - public
  rules:
    - if: $CI_COMMIT_REF_NAME == $CI_DEFAULT_BRANCH
```

**NOTA:** Prestar moita atención ás sangrías e espazos do documento anterior. Se non é correcta, produciranse erros. Pode comprobarse que a sintaxe do ficheiro anterior é correcta na páxina do proxecto en Gitlab -> **CI/CD** -> **Editor** -> **Lint**.

Poden subirse ao repositorio os cambios utilizando os comandos de git: add, commit e push. Cada vez que se fai un push, GitLab reconstrúe a páxina. Para ver como executa a páxina pódese acceder ao menú **CI/CD** -> **Pipelines**:



Se se pulsa no botón “**en ejecución**” -> **pages** vese como se van executando os diferentes pasos ata construír a web.

```

1 Running with gitlab-runner 13.8.0-rc1 (28e2e34a)
2   on docker-auto-scale 72989761
3   Preparing the "docker+machine" executor
4   Using Docker executor with image node:latest ...
5   Pulling docker image node:latest ...

```

E se non hai erros:

```

43 Saving cache for successful job
44 Creating cache default...
45 node_modules/: found 21559 matching files and directories
46 Uploading cache.zip to https://storage.googleapis.com/gitlab-com-runners-cache/project/23781624/default
47 Created cache
48 Uploading artifacts for successful job
49 Uploading artifacts...
50 public: found 16 matching files and directories
51 Uploading artifacts as "archive" to coordinator... ok id=973123781 responseStatus=201 Created token=NrR9-zoM
52 Cleaning up file based variables
53 Job succeeded

```

## Referencias

Para a elaboración deste material utilizáronse, entre outros, os recursos que se enumeran a continuación:

- [VuePress](#)
- [VuePress tutorial](#)
- [Introduction | VuePress](#)
- [GitLab Pages](#)
- [Create a Pages website from a forked sample | GitLab](#)
- [Generar sitios estáticos con VuePress y Vue.js – Styde.net](#)
- [Node.js, NPM, Yarn, NVM, Webpack, Babel... | by Mauricio Garcia](#)
- [El archivo package.json - Javascript en español - Lenguaje JS](#)
- [Static site generators: creación de proyectos web minimalistas](#)
- [GitHub Pages](#)
- [Deploying | VuePress](#)