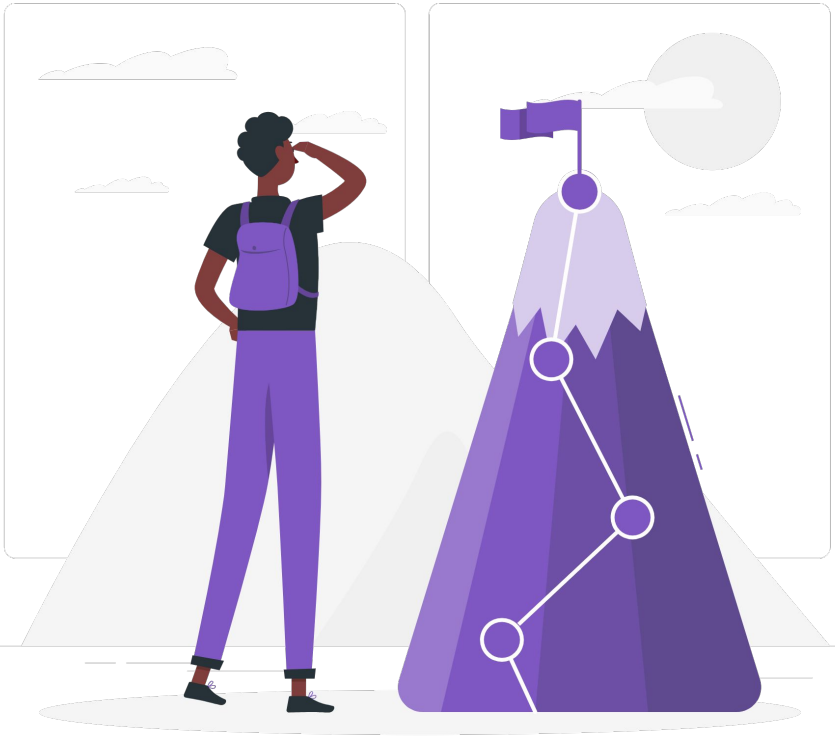
Estructura de Proyecto y Deployment

DEV.F:
DESARROLLAMOS(PERSONAS);

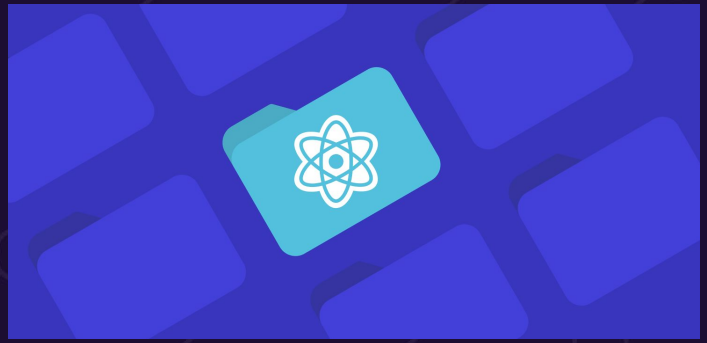
Elaborado por: César Guerra



Objetivos

- Aprender las mejores prácticas para estructurar proyectos de React.
- Aprender a preparar el código para producción
- Llevar a producción el código.

DEV.F::



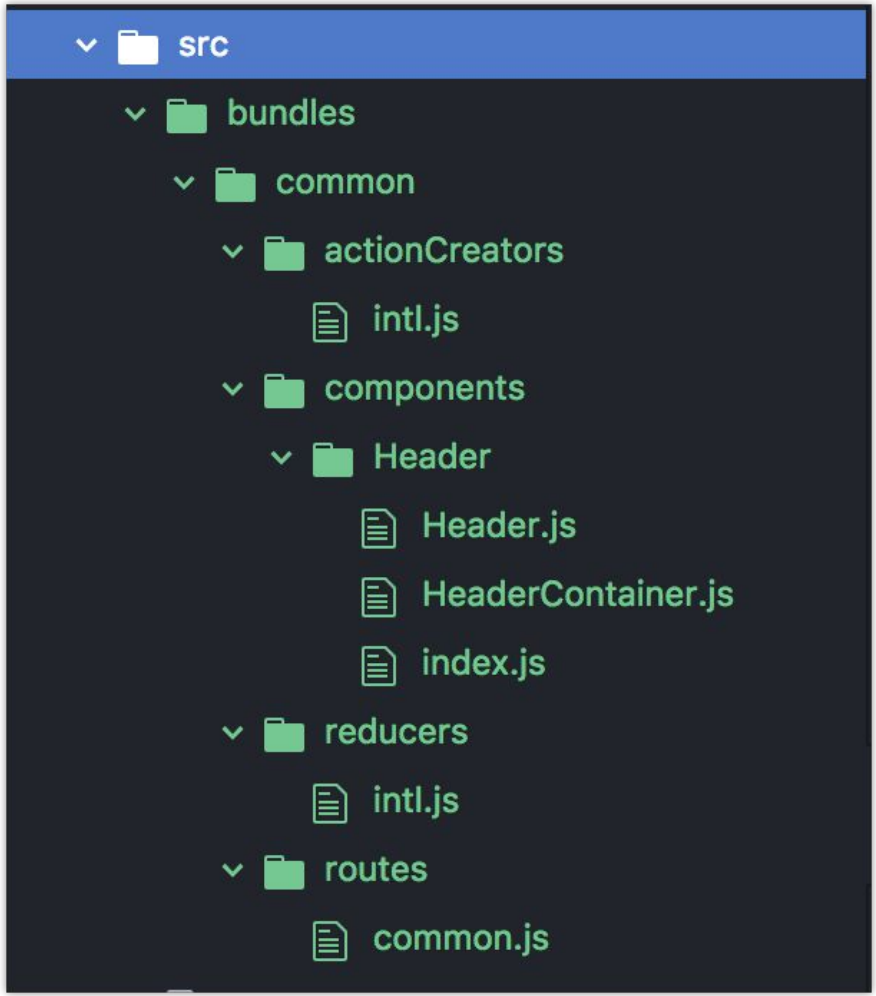
Estructura de Proyecto de React

DEV.F:: DESARROLLAMOS(PERSONAS);

DEV.F:

DISCUSIÓN

*¿Por qué debe preocuparnos **como organizar las carpetas/archivos** en nuestro proyecto de React?*



Problemática

Debido a la propia naturaleza de ReactJS **no existe una estructura de carpetas definida**, quedando a voluntad del desarrollador escoger la que más se adecua al proyecto que va a desarrollar.

Sin embargo, al no haber una estructura estándar a menudo surgen distintas formas de plantear esta organización, surgiendo a menudo la pregunta de si lo estaremos haciendo “bien” o por qué hacerlo en primer lugar.

DEV.F::

DEV.F

Hay que tener en cuenta qué ***no existe una metodología perfecta para estructurar un proyecto de React****, ya que depende de cada uno y de la naturaleza del proyecto la estructura a realizar.*

Sin embargo ***existen convenciones que podríamos usar como base****.*

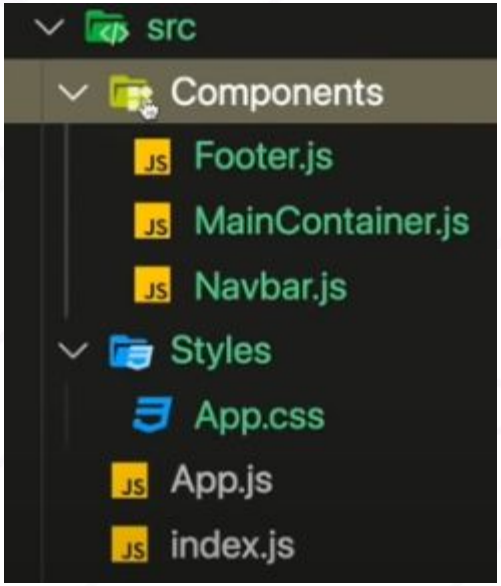
CONVENCIONES ÚTILES

De acuerdo al nivel de experiencia y tipo de proyecto

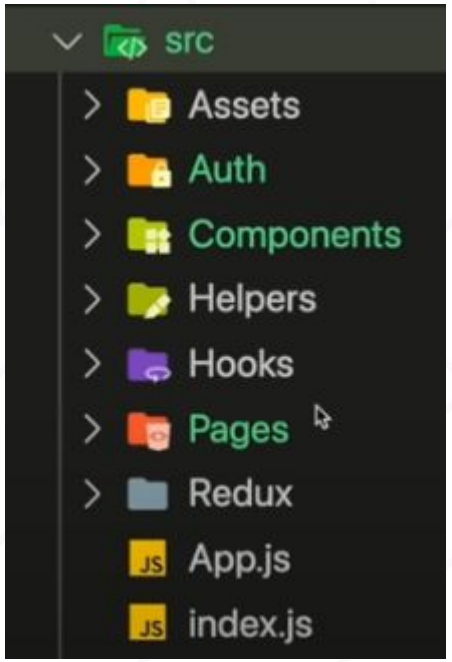
Conforme avanzamos en nuestro aprendizaje y creamos cada vez proyectos más complejos, la correcta organización del mismo nos facilitará mucho la tarea de seguir modificando el proyecto.

A continuación se presentan algunas convenciones de acuerdo al nivel de experiencia y tipo de proyecto.

Formas de Organizar un Proyecto



En proyectos pequeños o cuando comenzamos a aprender, mantenemos una estructura múy básica



En proyectos más complejos o cuando aprendemos más, comenzamos a estructurar mejor nuestro proyecto.

DEV.F

Carpeta: Components*

Los componentes son los bloques de construcción de cualquier proyecto react. Esta carpeta consiste en una colección de componentes de interfaz de usuario como botones, modales, entradas, cargador, etc., que pueden ser utilizados a través de varios archivos en el proyecto.

En aplicaciones más complejas, cada componente suele tener una carpeta, e incluso puede tener subcarpetas si hace uso de subcomponentes específicos para dicho componente.

Carpeta: Styles

Existen diversas formas de colocar estilos un proyecto de React, desde incluirlas directamente en cada componente a través de soluciones ***CSS-in-JS*** como ***Styled-Components***, pero en ocasiones queremos utilizar un archivo de estilos globales con CSS.

Es en esta carpeta donde colocaremos los archivos de estilos globales de la aplicación, ya sea en formato CSS, SASS ó SCSS.

Carpeta: Assets // Public

Esta carpeta contiene **todos los activos multimedia**, como imágenes, vídeos, archivos json, etc.

Carpeta: Layouts*

Como su nombre lo indica, contiene diseños disponibles y que pueden ser reutilizados para todo el proyecto o partes del mismo: como el encabezado, el pie de página, barras laterales, etc.

Carpeta: Hooks*

Esta carpeta contendrá todos los custom Hooks que usemos en el proyecto.

Recordemos que un custom hook es una función de JavaScript que hace uso de hooks nativos de react, pero no renderiza vistas, solo procesa información.

Un custom Hook debe poder ser reutilizable, por lo que podríamos llevarlos sin problemas a otros proyectos de React.

Carpeta: Pages*

Los archivos de la carpeta pages indican las vistas de la aplicación de react, por lo que **cada archivo de esta carpeta deberá contener su ruta**.

Una página puede contener subcarpetas de ser necesario.

Cada página tiene su estado y suele utilizarse para llamar a una operación asíncrona (ejemplo: llamar a una API).

Suele estar formada por varios componentes agrupados.

Carpeta: Routes***

Esta carpeta se colocan todas las rutas de la aplicación.

Consta de rutas de todo tipo:

- Públicas
- Privadas/Protegidas

Incluso podemos crear sub-rutas.

Por ejemplo, aquí es donde debemos implementar el ruteo con react router dom.

Carpeta: Services❌

Suele usarse para contener archivos de JavaScript con lógica de negocio relacionada a métodos de llamadas a la API externas.

De esta forma evitamos colocar esa lógica de negocio directamente en un componente y este siempre puede referenciar al servicio que necesite.

Carpeta: Utils

La carpeta ***Utils*** debe contener **funciones de JavaScript** que son usadas a largo de todo el proyecto para completar tareas abstractas de **forma genérica**.

Por ejemplo, funciones para dar formato a fechas, generar números aleatorios, dar formato a datos, regex, etc.

Estas funciones al ser tan genéricas, **podrían reutilizarse en otros proyectos.**

Carpeta: Helpers❌

Un **helper** es algo que nos ayuda a cumplir un propósito para un proyecto en específico, cuya lógica podemos reutilizar a lo largo del proyecto, pero qué no va ligado a un componente en específico.

Muchos desarrolladores tratan a la carpeta Helpers como sinónimo de **Utils**. Sin embargo existe una pequeña diferencia entre ambos:

Helpers son cosas específicas para el proyecto actual, mientras que **Utils** son funciones más genéricas que podrían reutilizarse en diversos proyectos.

Carpeta: Contexts*

La carpeta de ***Context*** es una carpeta que sólo contiene el estado que tiene que ser compartido a través de estos componentes.

Cada página puede tener varios contextos anidados, y cada contexto sólo pasa los datos hacia abajo. Pero para evitar la complejidad, es mejor tener sólo un archivo de contexto.

Carpeta: Config

Contiene los archivos de configuración especiales que hagan uso de variables de entorno (env)

Deployment

DEV.F:
DESARROLLAMOS(PERSONAS);

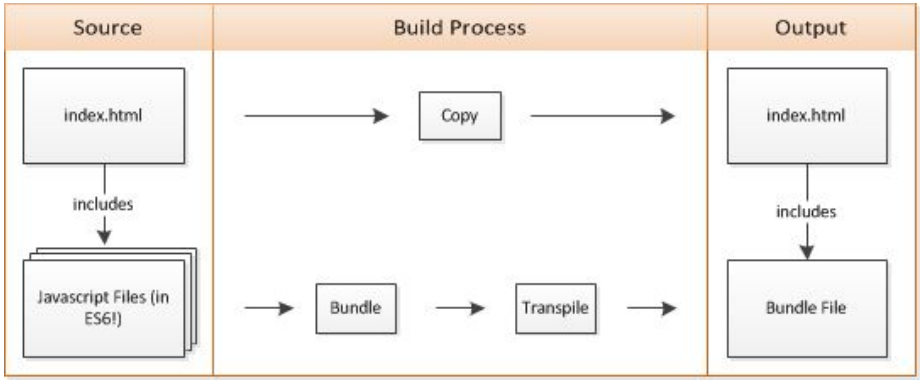
Deployment

Normalmente cuando escribimos el código y lo probamos en nuestro navegador estamos ejecutando un ambiente de desarrollo (development).

Un ***deployment*** consiste en hacer que una app esté disponible en un ***ambiente de producción*** o ***ambiente de prueba***.

El punto es que esté disponible para que usuarios o incluso otras apps o servicios puedan consumirlo.

DEV.F::



El proceso de build tiene varios propósitos pero no son obvios hasta que experimentas los problemas que se ocasionan cuando no implementas estas buenas prácticas.

Build

Antes de poder llevar nuestro código a producción, es necesario prepararlo, a esto se le conoce como hacer un ***build.*** En este proceso se realizan actividades como:

- Optimizar código para producción (transpilar, minificar, dividir).
- Crear y optimizar recursos (imágenes, fuentes, videos, etc).
- Eliminar código y dependencias de desarrollo.
- Identificar e instalar dependencias.
- Ejecutar pruebas.
- Versionar el código.

DEV.F::

Documentación Vite:

Building for Production | Vite (vitejs.dev)

Generar el código para Producción

Existen comandos para preparar nuestro código para producción. Dependerá mucho de qué estemos usando en nuestro proyecto de React.

Normalmente tanto para ***create-react-app*** como para ***Vite*** nos funciona:

npm run build

DEV.F::

```
$ npm run build

> global-state@0.1.0 build D:\warde\Documents\_Sites\2020\MASTER-EN-CODE
ate
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

  41.75 KB  build\static\js\2.043d8c4d.chunk.js
  1.81 KB   build\static\js\main.08e9bc68.chunk.js
  1.4 KB    build\static\js\3.4cecef72.chunk.js
  1.17 KB   build\static\js\runtime-main.1ce9c7c5.js
  684 B     build\static\css\main.fbb28094.chunk.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

  npm install -g serve
  serve -s build

Find out more about deployment here:

  https://cra.link/deployment
```

CRA

```
$ npm run build

> deploy-test@0.0.0 build
> vite build

vite v2.9.6 building for production...
✓ 34 modules transformed.
dist/assets/favicon.17e50649.svg   1.49 KiB
dist/assets/logo.ecc203fb.svg      2.61 KiB
```

VITE

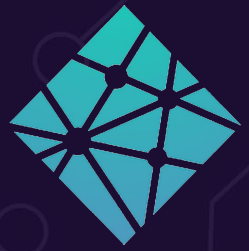
Resultado de un Build

Ejecutar un build, generará una nueva carpeta en nuestro proyecto con los archivos procesados y listos para enviar a producción.

Normalmente las carpetas con un build suelen generarse por defecto con alguno de estos nombres:

- **build** (create-react-app)
- **dist** (vite)

DEV.F::



netlify

Deployment en Netlify

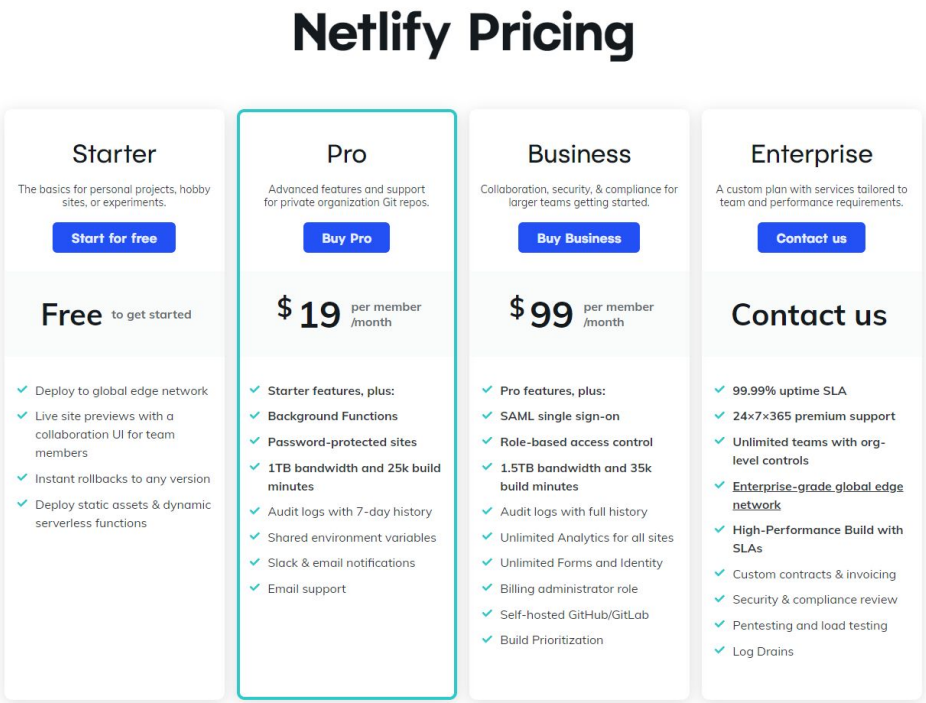
DEV.F:: DESARROLLAMOS(PERSONAS);

¿Qué es Netlify?

Netlify es una empresa de computación en la nube con sede en San Francisco que nace en el 2014 y ofrece servicios de alojamiento y backend sin servidor para aplicaciones web y sitios web estáticos.

DEV.F::

Precios Netlify



https://www.netlify.com/pricing/

DEV.F

DEV.F:

DISCUSIÓN

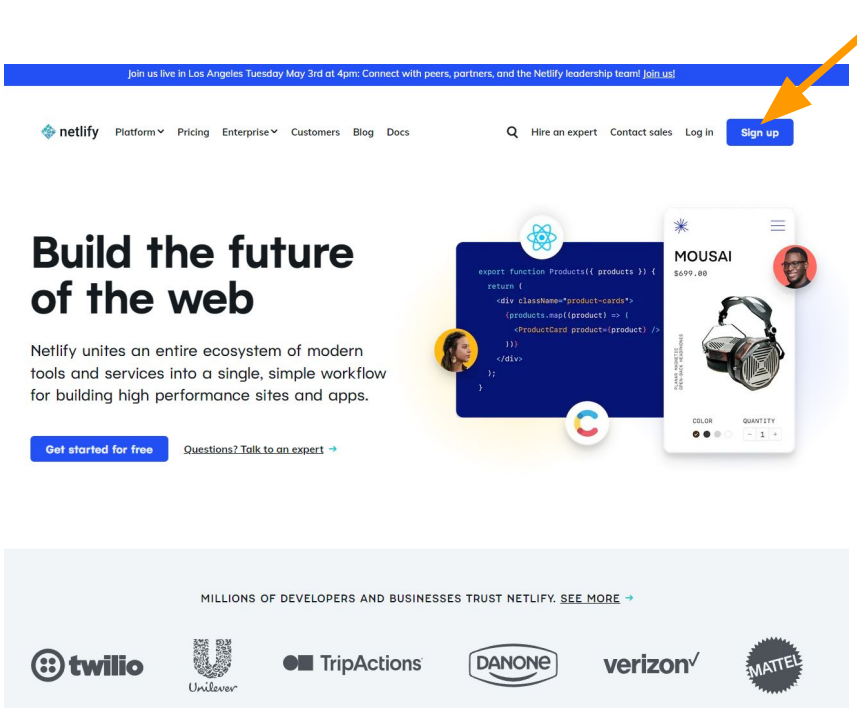
*¿Por qué pagar por servicios de hospedaje **netlify** en vez de nosotros montar nuestro propio servidor?*

Hágalo usted mismo vs pagar por el servicio

Usar tu compu para hostear tu app gratis

Se cae tu app cuando apagas la compu

imgflip.com



https://www.netlify.com/

Crear una cuenta en Netlify

El primer paso, es tener una cuenta en Netlify.

Podemos crearla con nuestro correo o usar alguna de las opciones que nos ofrece (cómo usar la cuenta gmail, github, etc.).

Esta cuenta nos permitirá almacenar proyectos de sitios en producción.

DEV.F::

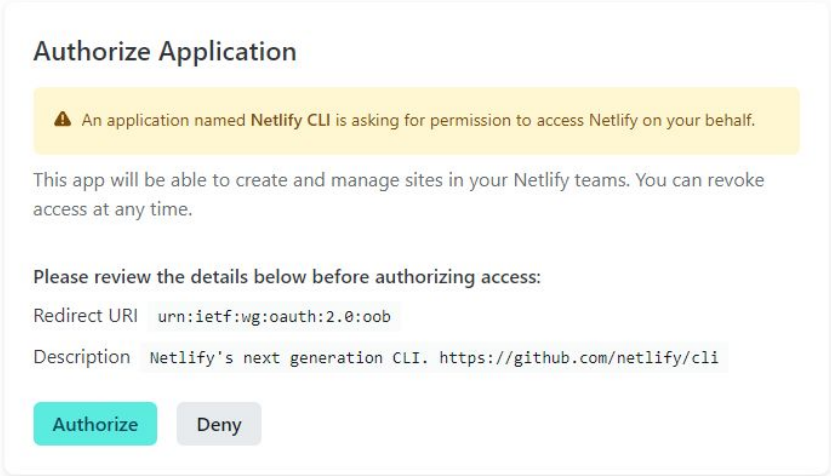
Instalar netlify-cli

Para llevar nuestro sitio a netlify de forma manual, debemos instalar de forma global una dependencia que nos permitirá llevar nuestro código a producción.

```
npm install netlify-cli -g
```

```
Success! Netlify CLI has been installed!

Your device is now configured to use Netlify CLI to deploy and manage your Netlify sites.

Next steps:

  netlify init     Connect or create a Netlify site from current directory
  netlify deploy   Deploy the latest changes to your Netlify site

For more information on the CLI run netlify help
Or visit the docs at https://cli.netlify.com
```

DEV.F::



Netlify deploy

Una vez instalado ***netlify-cli***, desde la carpeta del proyecto ejecutaremos el comando netlify deploy.

La primera vez que lo hagamos, se abrirá nuestro navegador, donde nos pedirá que iniciemos sesión y posteriormente autorizar a la aplicación.

Esto solo lo hace la primera vez que usamos ***netlify deploy***.

DEV.F::

```
You are now logged into your Netlify account!

Run netlify status for account details

To see all available commands run: netlify help

This folder isn't linked to a site yet
? What would you like to do? +  Create & configure a new site
? Team: César Guerra's team
Choose a unique site name (e.g. warderer-makes-great-sites.netlif
y.app) or leave it blank for a random name. You can update the si
te name later.
? Site name (optional): deploy-test-01

Site Created

Admin URL: https://app.netlify.com/sites/deploy-test-01
URL:       https://deploy-test-01.netlify.app
Site ID:   273264bd-35df-4e83-84db-80d6f2f9d8b4
```

Configurando el proyecto de Netlify

Una vez autenticados, el asistente nos va a preguntar:

What would you like to do?: Decir si es un sitio nuevo o uno existente. La primera vez seleccionaremos ***Create & configure a new site***

Team: Decir que equipo (de nuestra cuenta de netlify) se va a usar. Usar el valor por defecto.

Site name (opcional): El nombre proyecto para que se genere el nombre de la url.

Publish Directory: Indicamos como se llama la carpeta que contiene el código para producción, si usamos **vite** colocaremos: ***dist***

DEV.F::

```
Website Draft URL: https://6269b2b9068b57076e0d6284--deploy-test-
01.netlify.app

If everything looks good on your draft URL, deploy it to your mai
n site URL with the --prod flag.
netlify deploy --prod
```


```
$ netlify deploy --prod
Please provide a publish directory (e.g. "public" or "dist" or ".
"):
D:\warderer\Documents\GitHub\2022\test\deploy-test
? Publish directory D:\warderer\Documents\GitHub\2022\test\deploy
-test\dist
Deploy path: D:\warderer\Documents\GitHub\2022\test\deploy-test\d
ist
Deploying to main site URL...
✓ Finished hashing
✓ CDN requesting 0 files
✓ Finished uploading 0 assets
✓ Deploy is live!

Logs:              https://app.netlify.com/sites/deploy-test-01/d
eploys/6269b68e61d7220a04425f5f
Unique Deploy URL: https://6269b68e61d7220a04425f5f--deploy-test-
01.netlify.app
Website URL:       https://deploy-test-01.netlify.app
```

Configurando el proyecto de Netlify

Al final nos generará una URL de pruebas ***Website Draft URL***, donde podremos probar la aplicación, y si todo está en orden podremos entonces ahora sí mandarla a la URL de producción que nos proporcionó anteriormente, ejecutando el comando que nos indica.

DEV.F::