

Temas a tratar:

- principios de diseño en react
- renderizado condicional
- deploy en netlify

Principios de diseño

Los principios de diseño son las reglas que un equipo sigue para crear sus aplicaciones o proyectos, y el equipo de Facebook encargado de mantener y crear nuevas APIs para React no es la excepción, ellos también cuentan con ciertos principios a la hora de escribir su código. Los más importantes son:

Abstracciones comunes

En el core de React no se van a incluir APIs para resolver problemas demasiado específicos, a menos de que muchas personas estén resolviendo dicho problema de una manera muy mala y poco funcional.

Interoperabilidad

React debe ser capaz de convivir con otras herramientas sin causar ningún problema, por lo tanto podemos tener proyectos que usan diferentes herramientas y funcional igualmente, o también podemos migrar proyectos que están contruidos con otra herramienta y lentamente incluir React sin que la app se rompa.

Estabilidad

React no va a lanzar nuevas versiones de su core si estas rompen el código que ya existe, por lo tanto el código que escribimos hace varios meses o años todavía va a funcionar a día de hoy. En caso de que el equipo de React saque alguna funcionalidad que cause conflictos con APIs anteriores, entonces le avisaran a la comunidad para así poder implementar una solución diferente a tiempo.

Experiencia de desarrollo

React no solo busca que podamos implementar soluciones a nuestros problemas con sus APIs, sino que también busca que estas soluciones sean disfrutables, y que brinden una buena experiencia a la hora de ser implementadas.

Implementación

El código 'aburrido' siempre se va a preferir sobre el código elegante, ya que este último puede llegar a ser difícil de mover o eliminar en caso de que se deseen hacer modificaciones a la app.

Optimizado para la instrumentación

Los nombres de las APIs de react siempre tratarán de ser autodescriptivos, detallados y distintivos, aunque esto no significa que se vayan a colocar nombres muy largos.

Dogfooding

Debemos recordar que React es un proyecto creado por y para Facebook, por lo tanto el equipo encargado de crear nuevas funcionalidades siempre priorizará lo que necesita Facebook y no lo que quiere o necesita la comunidad. Aunque esto parece malo a primera vista, es todo lo contrario, ya que React cuenta con el apoyo y

mantenimiento de una empresa enorme, lo cual lo convierte en una herramienta muy confiable.

Planificación

Antes de empezar cualquier proyecto debemos definir qué responsabilidades le vamos a delegar a React y cuales dependen de nosotros.

Configuración

Nosotros no podemos, o mejor, no deberíamos modificar directamente el código de React, ya que esto podría causar conflictos a la hora de añadir nuevas herramientas al entorno de nuestro proyecto.

Depuración

React siempre nos va a dejar pequeñas pistas cada vez que ocurra un error dentro de nuestra aplicación, para así poder encontrar cuál es el foco del problema y saber que deberíamos cambiar.

Renderizado condicional

En React, puedes crear distintos componentes que encapsulan el comportamiento que necesitas. Entonces, puedes renderizar solamente algunos de ellos, dependiendo del estado de tu aplicación.

El renderizado condicional en React funciona de la misma forma que lo hacen las condiciones en JavaScript. Usa operadores de JavaScript como `if` o el operador condicional para crear elementos representando el estado actual, y deja que React actualice la interfaz de usuario para emparejarlos.

Considera estos dos componentes:

```
function UserGreeting(props) {  
  return <h1>Welcome back!</h1>;  
}  
  
function GuestGreeting(props) {  
  return <h1>Please sign up.</h1>;  
}
```

Vamos a crear un componente `Greeting` que muestra cualquiera de estos componentes dependiendo si el usuario ha iniciado sesión:

```
function Greeting(props) {  
  const isLoggedIn = props.isLoggedIn;  
  if (isLoggedIn) {  
    return <UserGreeting />;  
  }  
  return <GuestGreeting />;  
}
```

```
ReactDOM.render(  
  // Intentar cambiando isLoggedIn={true}:  
  <Greeting isLoggedIn={false} />,  
  document.getElementById('root')  
);
```

Este ejemplo renderiza un saludo diferente según el valor del prop `isLoggedIn`.

Variables de elementos

Puedes usar variables para almacenar elementos. Esto puede ayudarte para renderizar condicionalmente una parte del componente mientras el resto del resultado no cambia.

Considera estos dos componentes nuevos que representan botones de cierre e inicio de sesión:

```
function LoginButton(props) {  
  return (  
    <button onClick={props.onClick}>  
      Login  
    </button>  
  );  
}  
  
function LogoutButton(props) {  
  return (  
    <button onClick={props.onClick}>  
      Logout  
    </button>  
  );  
}
```

En el siguiente ejemplo, crearemos un componente con estado llamado LoginControl.

El componente va a renderizar <LoginButton /> o <LogoutButton /> dependiendo de su estado actual. También va a renderizar un <Greeting /> del ejemplo anterior:

```
class LoginControl extends React.Component {  
  constructor(props) {  
    super(props);  
  
    this.handleLoginClick = this.handleLoginClick.bind(this);  
    this.handleLogoutClick = this.handleLogoutClick.bind(this);  
  
    this.state = {isLoggedIn: false};  
  }  
  
  handleLoginClick() {  
    this.setState({isLoggedIn: true});  
  }  
}
```

```

handleLogoutClick() {
  this.setState({isLoggedIn: false});
}

render() {
  const isLoggedIn = this.state.isLoggedIn;

  let button;

  if (isLoggedIn) {
    button = <LogoutButton onClick={this.handleLogoutClick} />;
  } else {
    button = <LoginButton onClick={this.handleLoginClick} />;
  }

  return (
    <div>
      <Greeting isLoggedIn={isLoggedIn} />
      {button}
    </div>
  );
}
}

ReactDOM.render(
  <LoginControl />,
  document.getElementById('root')
);

```

Pruébalo en CodePen

Si bien declarar una variable y usar una sentencia if es una buena forma de renderizar condicionalmente un componente, a veces podrías querer usar una sintaxis más corta. Hay algunas formas de hacer condiciones en una línea en JSX, explicadas a continuación.

If en una línea con operador lógico &&

Puedes incluir expresiones en JSX envolviéndolas en llaves. Esto incluye el operador lógico && de JavaScript. Puede ser útil para incluir condicionalmente un elemento:

```
function Mailbox(props) {
  const unreadMessages = props.unreadMessages;
  return (
    <div>
      <h1>Hello!</h1>
      {unreadMessages.length > 0 &&
        <h2>
          You have {unreadMessages.length} unread messages.
        </h2>
      }
    </div>
  );
}
```

```
const messages = ['React', 'Re: React', 'Re:Re: React'];
ReactDOM.render(
  <Mailbox unreadMessages={messages} />,
  document.getElementById('root')
);
```

Esto funciona porque en JavaScript, `true && expresión` siempre evalúa a `expresión`, y `false && expresión` siempre evalúa a `false`.

Por eso, si la condición es `true`, el elemento justo después de `&&` aparecerá en el resultado. Si es `false`, React lo ignorará.

Ten en cuenta que retornar expresiones falsas hará que el elemento después de `'&&'` sea omitido pero retornará el valor falso. En el ejemplo de abajo, `'0'` será retornado por el método de renderizado.

```
render() {
  const count = 0;
  return (
    <div>
```

```

    {count} && <h1>Messages: {count}</h1>}
  </div>
);
}

```

If-Else en una línea con operador condicional

Otro método para el renderizado condicional de elementos en una línea es usar el operador condicional condición ? true : false de JavaScript.

En el siguiente ejemplo, lo usaremos para renderizar de forma condicional un pequeño bloque de texto.

```

render() {
  const isLoggedIn = this.state.isLoggedIn;
  return (
    <div>
      The user is <b>{isLoggedIn ? 'currently' : 'not'}</b> logged in.
    </div>
  );
}

```

También puede usarse para expresiones más grandes, aunque es menos obvio lo que está pasando:

```

render() {
  const isLoggedIn = this.state.isLoggedIn;
  return (
    <div>
      {isLoggedIn
        ? <LogoutButton onClick={this.handleLogoutClick} />
        : <LoginButton onClick={this.handleLoginClick} />
      }
    </div>
  );
}

```


Al igual que en JavaScript, depende de ti elegir un estilo apropiado según lo que tú y tu equipo consideren más legible. Recuerda también que cuando las condiciones se vuelven demasiado complejas, puede ser un buen momento para extraer un componente.

Evitar que el componente se renderice

En casos excepcionales, es posible que desees que un componente se oculte a sí mismo aunque haya sido renderizado por otro componente. Para hacer esto, devuelve null en lugar del resultado de renderizado.

En el siguiente ejemplo, el `<WarningBanner />` se renderiza dependiendo del valor del prop llamado `warn`. Si el valor del prop es `false`, entonces el componente no se renderiza:

```
function WarningBanner(props) {  
  if (!props.warn) {  
    return null;  
  }  
  return (  
    <div className="warning">  
      Warning!  
    </div>  
  );  
}
```

```
class Page extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {showWarning: true};  
    this.handleClick = this.handleClick.bind(this);  
  }  
  handleClick() {  
    this.setState(state => ({  
      showWarning: !state.showWarning  
    }));  
  }  
}
```

```
render() {  
  return (  
    <div>  
      <WarningBanner warn={this.state.showWarning} />  
      <button onClick={this.handleToggleClick}>  
        {this.state.showWarning ? 'Hide' : 'Show'}  
      </button>  
    </div>  
  );  
}
```

```
ReactDOM.render(  
  <Page />,  
  document.getElementById('root')  
);
```

El devolver null desde el método render de un componente no influye en la activación de los métodos del ciclo de vida del componente. Por ejemplo `componentDidUpdate` seguirá siendo llamado.

¿Qué es netlify?



Netlify es una plataforma que nace para automatizar proyectos webs estáticos. Aúna las tareas de integración continua y despliegue de infraestructura web en un solo flujo de ejecución

El desarrollo web se caracteriza por, entre otras cosas, el cambio continuo en los diseños. Constantemente se añaden nuevos elementos o se modifican los ya añadidos. Es muy importante poder ver en todo momento qué aspecto tiene o ha tenido nuestra web.

Otra de las claves es la gestión de la infraestructura. No sólo vale centrarse en el diseño de la web, si no que también esta debe prepararse para ser ejecutada en un entorno de producción determinado. Esta tarea aparentemente sencilla, a menudo se convierte en un proceso tedioso: debemos contratar un hosting donde alojar la web, registrar un nombre de dominio y finalmente subir los archivos, normalmente a través de FTP.

Con Netlify el proceso de despliegue se convierte en algo muy sencillo: únicamente hay que enlazar la herramienta a un repositorio Git donde se encuentren los archivos que componen la página web y crear un deploy que provocará que la aplicación se compile y se despliegue automáticamente en una determinada URL.

Su potencia viene dada por su capacidad de despliegue continuo. Todos los cambios que se hayan realizado en la aplicación web constituyen versiones desplegadas de la misma, a las que se puede tener acceso en cualquier momento. Si la versión actual de la web no nos convence, podemos dejarla en un estado en el que se encontraba anteriormente.

Despliegue continuo

Tras tener listo el código en el repositorio correspondiente (github), creamos una cuenta gratuita en Netlify para poder tener acceso a sus herramientas y hacer un despliegue continuo de nuestro proyecto. Existen distintos planes de pago con distintas características que se pueden consultar aquí. En nuestro caso escogemos el gratuito, ya que se adapta de manera holgada al objetivo del tutorial.

Siguiendo con el desarrollo, crear la cuenta es tan fácil como registrarnos con nuestra cuenta de GitHub, GitLab, Bitbucket o con nuestro correo. Debemos dar permisos a Netlify para acceder a nuestra cuenta GitHub e introducir el nombre de usuario y el tipo de proyecto que se va a desplegar: un portfolio, una tienda virtual, un blog, etc.



Welcome to Netlify

Log in with one of the following:

 GitHub

 GitLab

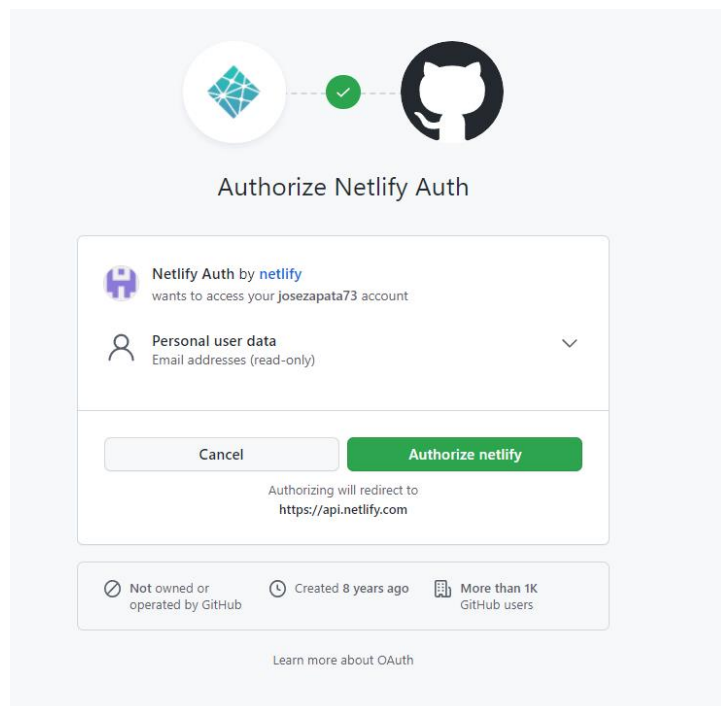
 Bitbucket

Email

[Log in via SSO](#)

No account yet? [Sign up](#)

Luego debemos dar la autorización a netlify:



Seguimos con otras configuraciones del tipo de cuenta:

Get started

Tell us about yourself

Your responses will help us improve your experience.

I'm using Netlify for

Required field

I'm a:

My first project will be:

Personalize your team space

Make your team your own.


Name your team

People who use Netlify for school often use a school, cohort or project name.

Luego nos pide que importemos un proyecto (recomendamos el proyecto que vienen trabajando en github), para poder vincularlo:


Deploy your first project

Import an existing project from a Git repository or pick a starter template.
Not ready yet? [Skip this step for now.](#)



Import an existing project

[Import from Git](#)



Start from a template

[Browse templates](#)

...or deploy manually

Drag and drop your site output folder here

Or, [browse to upload](#)

Luego nos va pidiendo una serie de pasos para vincular nuestra cuenta de netlify con los repositorios de github:

Import an existing project from a Git repository

Connect your Git repo, push your code... and that's it! Or [go back](#) to choose a different deploy method.

Step 1 of 3


Connect to Git provider


Choose the Git provider where your site's source code is hosted. When you push to Git, we run your build tool of choice on our servers and deploy the result.


You can [unlock options for self-hosted GitHub/GitLab](#) by upgrading to the Business plan.




Netlify by **Netlify** would like permission to:

 Verify your GitHub identity (josezapata73)

 Know which resources you can access

 Act on your behalf

Resources on your account

 **Email addresses** (read)
View your email addresses

Netlify has not been installed on any accounts you have access to.

[Learn more about Netlify](#)

Cancel

Authorize Netlify

Authorizing will redirect to
<https://api.netlify.com>



Install Netlify

Install on your personal account josezapata73



☒ **All repositories**

This applies to all future repositories.

with these permissions:

- ✓ **Read** access to code and metadata
- ✓ **Read and write** access to checks, commit statuses, issues, and pull requests

User permissions

Netlify can also request users' permission to the following resources. These permissions will be requested and authorized on an individual-user basis.

- ✓ **Read** access to emails

Install

[Cancel](#)

Next: you'll be directed to the GitHub App's site to complete setup.

Import an existing project from a Git repository

Connect your Git repo, push your code... and that's it! Or [go back](#) to choose a different deploy method.

Step 2 of 3

Pick a repository from GitHub

Choose the repository you want to link to your site on Netlify. When you push to Git, we run your build tool of choice on our servers and deploy the result.



josezapata73 ▾

🔍 Search repos



josezapata73/prueba-react-cac



Elegimos la rama en la cual esta nuestro proyecto y seleccionamos “Deploy Site”:

Import an existing project from a Git repository

Connect your Git repo, push your code... and that's it! Or [go back](#) to choose a different deploy method.

Step 3 of 3

Site settings for josezapata73/prueba-react-cac

Refine how Netlify builds and deploys your site with these settings.


Branch to deploy

master ▾

Customize build settings

Deploy site

Esperamos unos minutos hasta que haga el proceso:

 cac-react-2022 > spontaneous-otter-d7cfef

Upgrade

🔍 🔔 ⚙️ 🗄️

[Site overview](#) [Deploys](#) [Plugins](#) [Functions](#) [Edge Functions](#) [Identity](#) [Forms](#) [Large Media](#) [Split Testing](#) [Analytics](#) [Graph](#) [Site settings](#)


spontaneous-otter-d7cfef

• Site deploy in progress

Deploys from GitHub. Created at 6:32 PM.

⚙️ Site settings

⚙️ Domain settings



Getting started

1

Deploying your site ⚙️

Netlify's robots are busy building and deploying your site to our CDN.

2

Set up a custom domain

Buy a new domain or setup a domain you already own.

3

Secure your site with HTTPS

Your site is secured automatically with a Let's Encrypt certificate.

Deploy in progress for spontaneous-otter-d7cfef

⚙️ Netlify's robots are busy building and deploying your site to our CDN.

Today at 6:33 PM

Production: master@HEAD


Cancel deploy

⚙️ Deploy settings

Fancy a game while you wait?

We're on a mission to find the cutest emojis.

Play game



Deploy log

📄 ⬆️ ⬇️

1

6:33:30 PM: Waiting for other deploys from your team to complete. Check the queue: <https://app.netlify.com/teams/josezapata73/builds>

2

6:33:35 PM: Build ready to start

3

6:33:37 PM: build-image version: d2c6dbeac570350a387d832f64bc980dc964ad65 (focal)

4

6:33:37 PM: build-image tag: v4.8.0

5

6:33:37 PM: buildbot version: 13616ca0f4280331d49967e190a04e108291d30c

6

6:33:37 PM: Fetching cached dependencies

7

6:33:37 PM: Failed to fetch cache, continuing with build

8

6:33:37 PM: Starting to prepare the repo for build

9

6:33:37 PM: No cached dependencies found. Cloning fresh repo

10

6:33:37 PM: git clone <https://github.com/josezapata73/prueba-react-cac>

11

6:33:38 PM: Preparing Git Reference refs/heads/master

12

6:33:38 PM: Parsing package.json dependencies

13

6:33:39 PM: Starting build script


14

6:33:39 PM: Installing dependencies

15

6:33:39 PM: Python version set to 2.7

Una vez terminado el proceso, accedemos al link “permanlink” para ver nuestro proyecto

 cac-react-2022 > spontaneous-otter-d7cfef

Upgrade

🔍 🔔 ⚙️ 🗄️

[Deploys](#)

Published deploy for spontaneous-otter-d7cfef

[🔗 Permalink](#)

Today at 6:33 PM

Production: master@HEAD ⬇️

Open published deploy ➡️

Lock publishing to this deploy

Options ▾

Fancy a game?

This deploy is done, but hey, you can still play if you want to!

Play game

