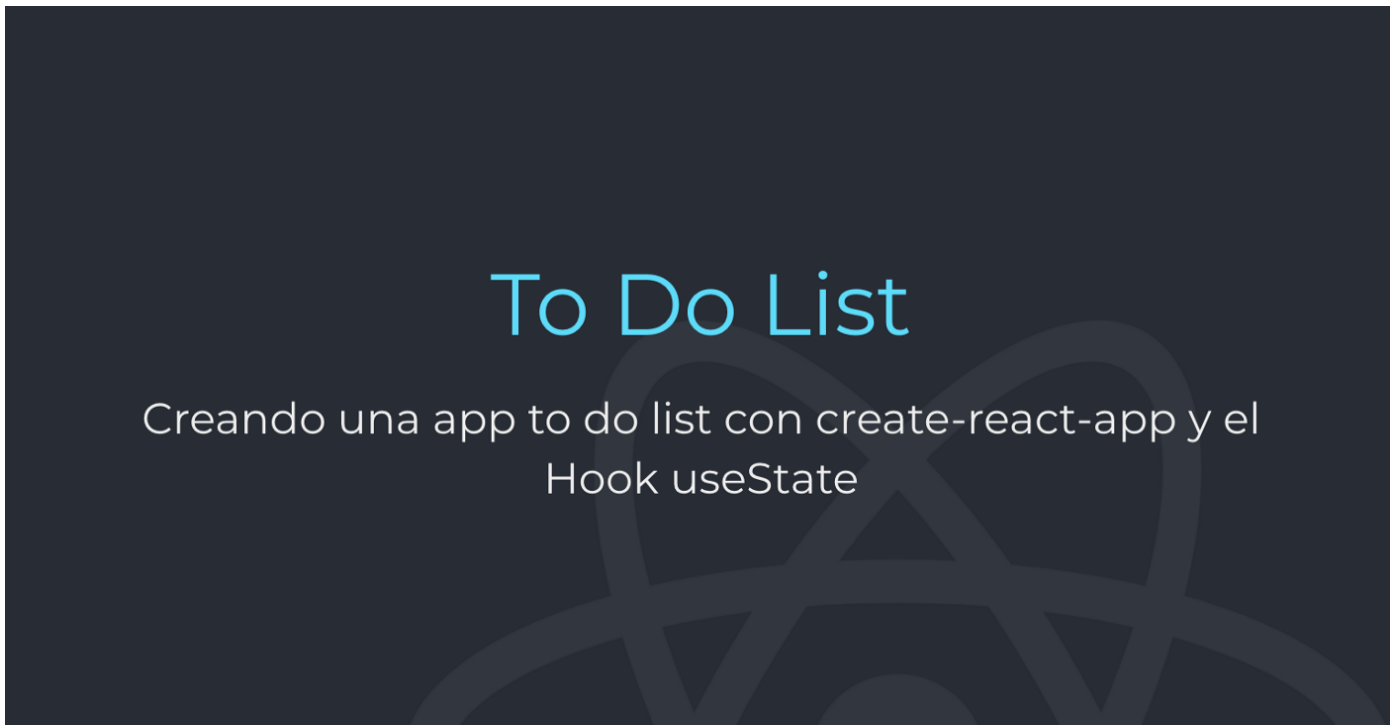


[Open in app](#)[Get started](#)Mauricio Garcia · [Follow](#)

Sep 7, 2020 · 8 min read



React — Creando una app to do list con create-react-app y el Hook useState



Temario

- Introducción
- Creando una app con create-react-app
- Definiendo los componentes
- Creando componentes funcionales
- Construyendo la aplicación para subir a producción
- Subiendo el proyecto a un servidor



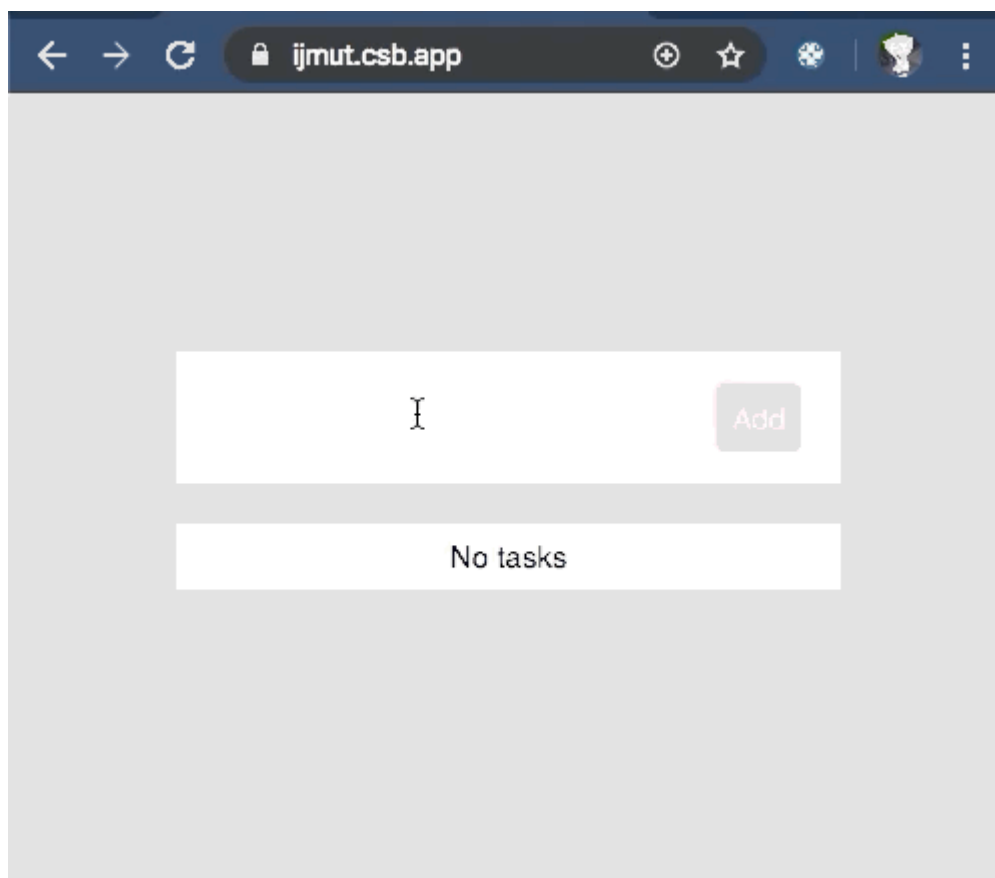
[Open in app](#)[Get started](#)

i. Introducción

En esta entrega vamos a **crear un to-do list** con **React**, lo haremos con **create-react-app**, **componentes funcionales** y el **Hook** `useState`.

Si llegaste aquí directamente y quieres **aprender React desde cero**, te recomiendo que inicies por acá (**React — Primeros pasos...**[\[ref\]](#))

La idea es tener algo así:



Ver proyecto en línea[\[ref\]](#)

Así que comencemos!!!!...



[Open in app](#)[Get started](#)

```
1 npx create-react-app my-todo-list-app
```

command.txt hosted with ❤ by GitHub

[view raw](#)

Si tienes dudas por que npx o llegaste aquí directo, te recomiendo que inicies primero por acá (Mi primera App con create-react-app [ref])

Vamos a instalar el módulo para poder usar **SASS**, entonces en consola ejecutamos el siguiente comando:

```
1 //NPM
2 npm install node-sass
3
4 //Yarn
5 yarn add node-sass
```

command-3.txt hosted with ❤ by GitHub

[view raw](#)

Una vez que hemos descargado todo, vamos a levantar el servidor, y ver nuestra aplicación, pueden usar yarn o npm :

```
1 //NPM
2 npm start
3
4 //Yarn
5 yarn start
```

command-2.txt hosted with ❤ by GitHub

[view raw](#)

Se debe **abrir** una **ventana** en el **navegador**, mostrando la **aplicación**.

iii. Definiendo los componentes

El siguiente paso, es **definir** qué **componentes** son los que vamos a **construir**, entonces:





Open in app

Get started

Container

FormTodo

TaskList

Checkbox

- Checkbox — Es el **componente** que va a tener lo necesario para **mostrar el checkbox**, poder **marcar/desmarcar** una **tarea**
- FormTodo — Es el **componente** donde vamos a **escribir** una **tarea**, y vamos a poder **agregarla**.
- TaskList — Es el **componente** que va a tener una **lista** de **tareas** por hacer o hechas, donde va a **poder eliminar** todas las **tareas** que ya estén **hechas**.
- Container — Es el **componente** que va a **pasar** FormTodo a TaskList

iv. Creando componentes funcionales

Antes de generar los **componentes funcionales**, primero, vamos a **limpiar** el **proyecto**:

El componente `App.js` debe quedar así:

```
1 import React from 'react';
```



[Open in app](#)[Get started](#)

```
7  
8 export default App;
```

react-app-todoApp.js hosted with ❤ by GitHub

[view raw](#)

Los **estilos** App.css **cambiar el nombre** a App.scss y debe quedar así:

```
1  html {  
2    background: #ddd;  
3    height: 100%;  
4    display: flex;  
5  }  
6  body {  
7    width: 100%;  
8    margin: auto;  
9  }  
10  
11  .App {  
12    font-family: sans-serif;  
13    text-align: center;  
14  }
```

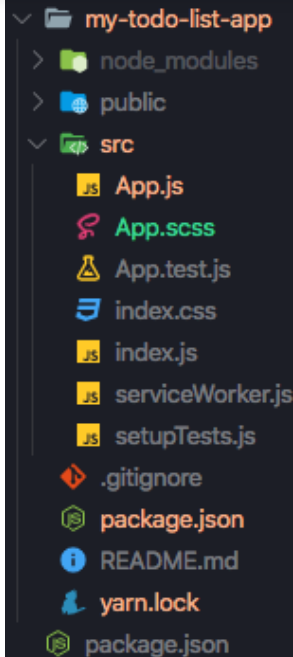
react-app-todoApp.scss hosted with ❤ by GitHub

[view raw](#)

Eliminar : logo.svg

Nuestra carpeta debe verse así:



[Open in app](#)[Get started](#)

Ahora si el **siguiente paso**, es ir **creando** uno a uno de los **componentes funcionales**, pero sin nada de código (cargando sus respectivos hijos)

- Dentro de la **carpeta** `src` debemos **crear** la **carpeta** `components`
- Dentro de la **carpeta** `components`, vamos a crear el **componente funcional** `Container.jsx` (si estás utilizando visual code e instalaste los plugins, solo basta escribir *rafce* (react arrow function componente export)) y dar enter.

```
1  import React from 'react'
2
3  const Container = () => {
4    return (
5      <div>
6        Container!
7      </div>
8    )
9  }
10
11  export default Container
```



[Open in app](#)[Get started](#)

```
1  // ...
2  import Container from './components/Container'
3
4  function App() {
5    return (
6      <div className="App">
7        <Container />
8      </div>
9    );
10 }
11
12 // ...
```

react-app-todo-App-2.js hosted with ❤ by GitHub

[view raw](#)

• Creamos el componente FormTodo.jsx

```
1  import React from "react";
2
3  const FormTodo = () => {
4    return <div>FormTodo!</div>;
5  };
6
7  export default FormTodo;
```

react-app-todo-FormTodo-1.jsx hosted with ❤ by GitHub

[view raw](#)

• Creamos el componente TaskList.jsx

```
1  import React from "react";
2
3  const TaskList = () => {
4    return <div>TaskList!</div>;
5  };
6
7  export default TaskList;
```

react-app-todo-TaskList-1.jsx hosted with ❤ by GitHub

[view raw](#)

[Open in app](#)[Get started](#)

```
1  import React from "react";
2
3  import FormTodo from "./FormTodo";
4  import TaskList from "./TaskList";
5
6  const Container = () => {
7    return (
8      <div>
9        Container!
10       <FormTodo />
11       <TaskList />
12     </div>
13   );
14 };
15
16 export default Container;
```

react-app-todo-Container-2.jsx hosted with ❤ by GitHub

[view raw](#)

- **Creamos el componente** Checkbox.jsx

```
1  import React from "react";
2
3  const Checkbox = () => {
4    return <div>Checkbox!</div>;
5  };
6
7  export default Checkbox;
```

react-app-todo-Checkbox-1.jsx hosted with ❤ by GitHub

[view raw](#)

- **Vamos a importar y agregar al JSX** Checkbox **en el componente funcional** TaskList

```
1  // ...
2  import Checkbox from "./Checkbox";
3
4  const TaskList = () => {
5    return (
```



[Open in app](#)[Get started](#)

```
11  };  
12  // ...
```

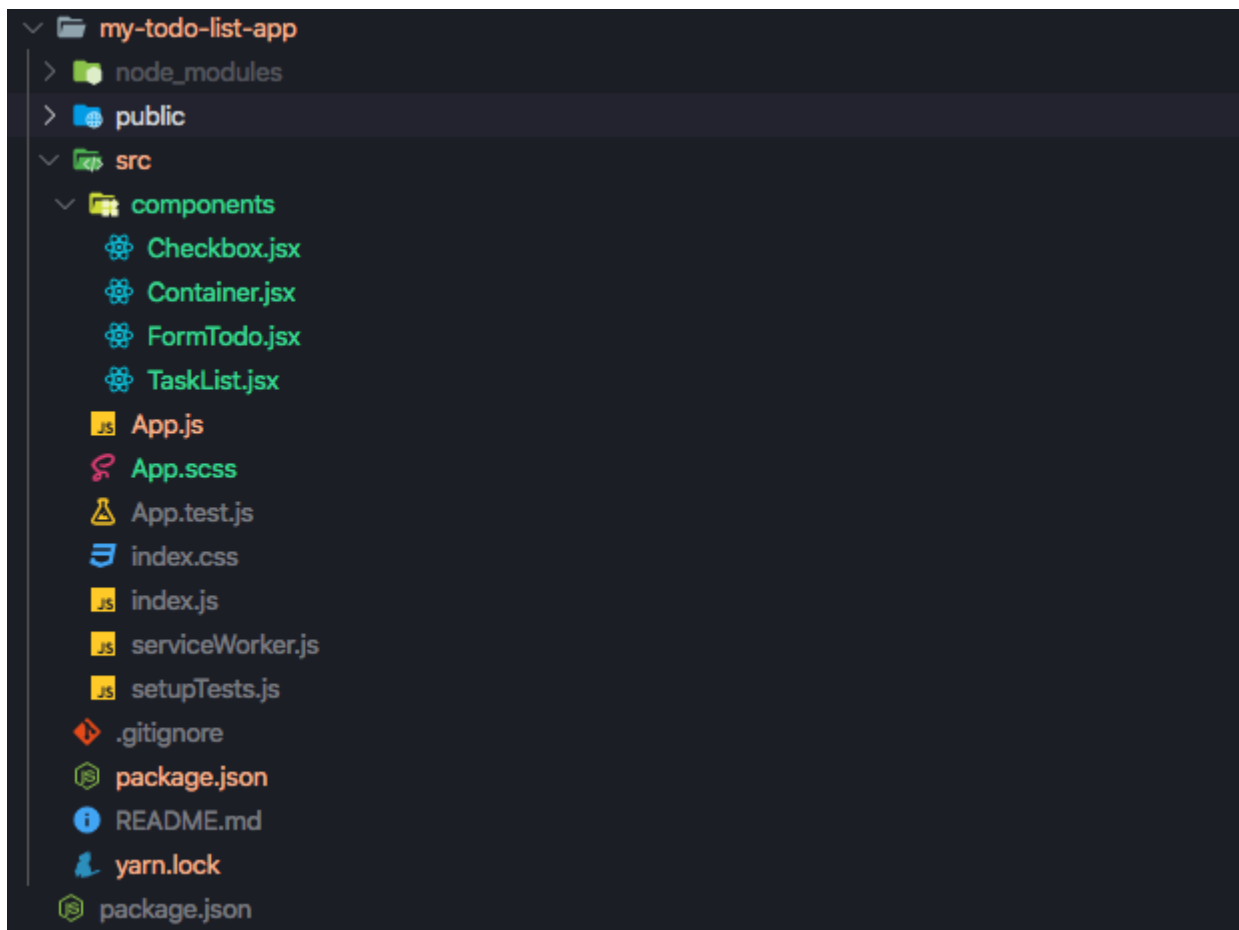
react-app-todo-TaskList-2.jsx hosted with ❤ by GitHub

[view raw](#)

Si lo hemos hecho todo bien, debemos de ver lo siguiente en el navegador:

Container!
FormTodo!
TaskList!
Checkbox!

Y nuestras carpetas deben verse así:

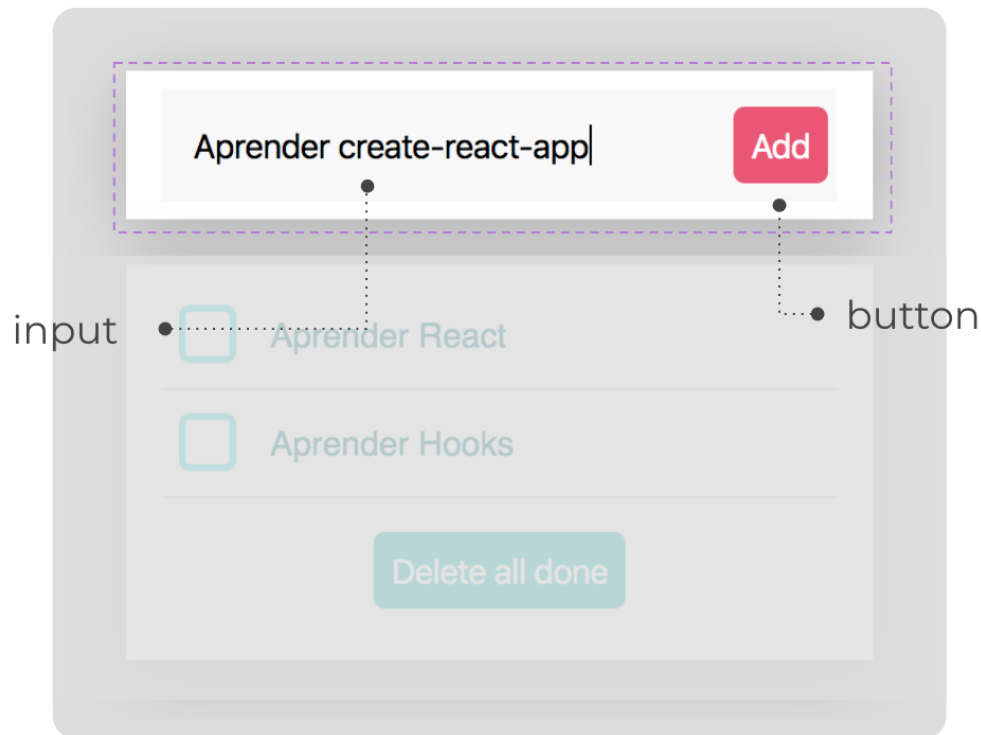




Open in app

Get started

FormTodo



:: Vamos paso a paso

- Primero **agregamos** los **elementos** `input` y `button`.
- Después **generamos** el **estado** `description` para **asignarlo** al `input`.
- Finalmente **validamos** el **botón**, si `description` no tiene valor lo **deshabilitamos**, caso contrario lo **habilitamos**.

```

1  // ...
2  const FormTodo = props => {
3    const [description, setDescription] = useState("");
4    return (
5      <form>
6        <div className="todo-list">
7          <div className="file-input">
8            <input
9              type="text"
10             className="text"

```




[Open in app](#)
[Get started](#)

```

15         className="button pink
16         disabled={description ? "" : "disabled"}
17     >
18         Add
19     </button>
20 </div>
21 </div>
22 </form>
23 );
24 };
25 // ...

```

FormTodo-1-1.jsx hosted with ❤ by GitHub

[view raw](#)

- Vamos a crear un manejador de evento (`onSubmit`) para el formulario `form` .
- Cuando se de **enter** en el `input` o **click** en el **botón**, debe **ejecutar el método** `handleSubmit` (**No olvidemos usar** `e.preventDefault()` , para **evitar** que se **refresque la página**).
- Por último, **limpiamos el estado** `description`, con `setDescription` .

```

1 // ...
2 const FormTodo = props => {
3     const [description, setDescription] = useState("");
4     const handleSubmit = e => {
5         e.preventDefault();
6         console.log(description);
7         setDescription("");
8     };
9     return (
10         <form onSubmit={handleSubmit}>
11             // ...
12         </form>
13     );
14 };
15 // ...

```

FormTodo-1-2.jsx hosted with ❤ by GitHub

[view raw](#)


[Open in app](#)[Get started](#)

```
1 // ...
2 const FormTodo = props => {
3   // ...
4   const { handleAddItem } = props;
5   const handleSubmit = e => {
6     e.preventDefault();
7
8     handleAddItem({
9       done: false,
10      id: (+new Date()).toString(),
11      description
12    });
13    setDescription("");
14  };
15  return // ...
16 };
17 // ...
```

FormTodo-1-3.jsx hosted with ❤ by GitHub

[view raw](#)

:: Código completo

```
1 import React, { useState } from "react";
2
3 const FormTodo = props => {
4   const { handleAddItem } = props; //(C-1)
5   const [description, setDescription] = useState(""); // (F-1)
6   const handleSubmit = e => {
7     e.preventDefault(); // (E)
8     // (C-2)
9     handleAddItem({
10      done: false,
11      id: (+new Date()).toString(),
12      description
13    });
14    setDescription(""); // (B)
15  };
16  return (
17    <form onSubmit={handleSubmit}>
```



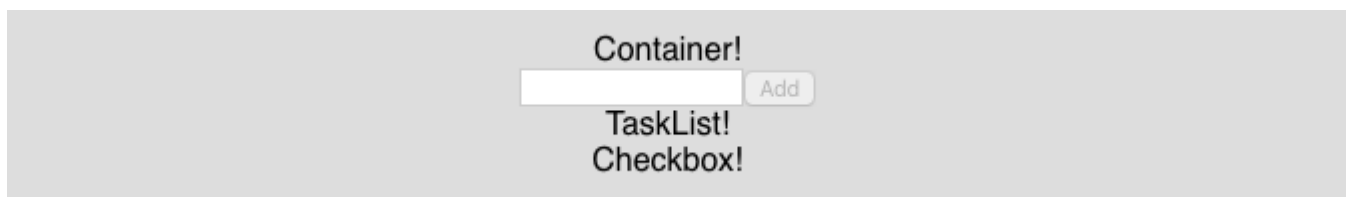
[Open in app](#)[Get started](#)

```
23         type="text"
24         className="text"
25         value={description}
26         onChange={e => setDescription(e.target.value)}
27     />
28     { /* (A) */ }
29     <button
30         className="button pink"
31         disabled={description ? "" : "disabled"}
32     >
33         Add
34     </button>
35 </div>
36 </div>
37 </form>
38 );
39 };
40
41 export default FormTodo;
```

react-app-todo-FormTodo-2.jsx hosted with ♥ by GitHub

[view raw](#)

:: Ejemplo de la app



b — Container

Vamos a darle forma al **componente funcional** `Container`, para que quede de la siguiente manera:



[Open in app](#)[Get started](#)

Container



:: Vamos paso a paso

- Vamos a **crear** el **estado** `list`, que nos va a servir para **almacenar** los **datos**.
- En el **componente** `FormTodo` vamos a **agregarle** el **atributo** `handleAddItem` y le vamos a **pasar** el **método** `handleAddItem`, donde vamos a **actualizar** el **estado** `list`.

```
1 // ...
2 const Container = () => {
3   const [list, setList] = useState([]);
4
5   const handleAddItem = addItem => {
6     setList([...list, addItem]);
7   };
8   return (
9     <div>
10       <FormTodo handleAddItem={handleAddItem} />
11       <TaskList/>
12     </div>
```



[Open in app](#)[Get started](#)

- Agregamos los atributos de `list` y `setList` en el componente `TaskList`, y le pasamos el estado.

```
1 // ...
2 const Container = () => {
3   // ...
4   return (
5     <div>
6       // ...
7       <TaskList list={list} setList={setList} />
8     </div>
9   );
10 };
11 // ...
```

Container-1-2.jsx hosted with ♥ by GitHub

[view raw](#)

:: Código completo

```
1 import React, { useState } from "react";
2
3 import TaskList from "../TaskList";
4 import FormTodo from "../FormTodo";
5
6 const Container = () => {
7   const [list, setList] = useState([]); // (B-1)
8
9   // (A-2)
10  const handleAddItem = addItem => {
11    setList([...list, addItem]); // (B-2)
12  };
13  return (
14    <div>
15      /*(A-1)*/
16      <FormTodo handleAddItem={handleAddItem} />
17      /*(C)*/
18      <TaskList list={list} setList={setList} />
19    </div>
```



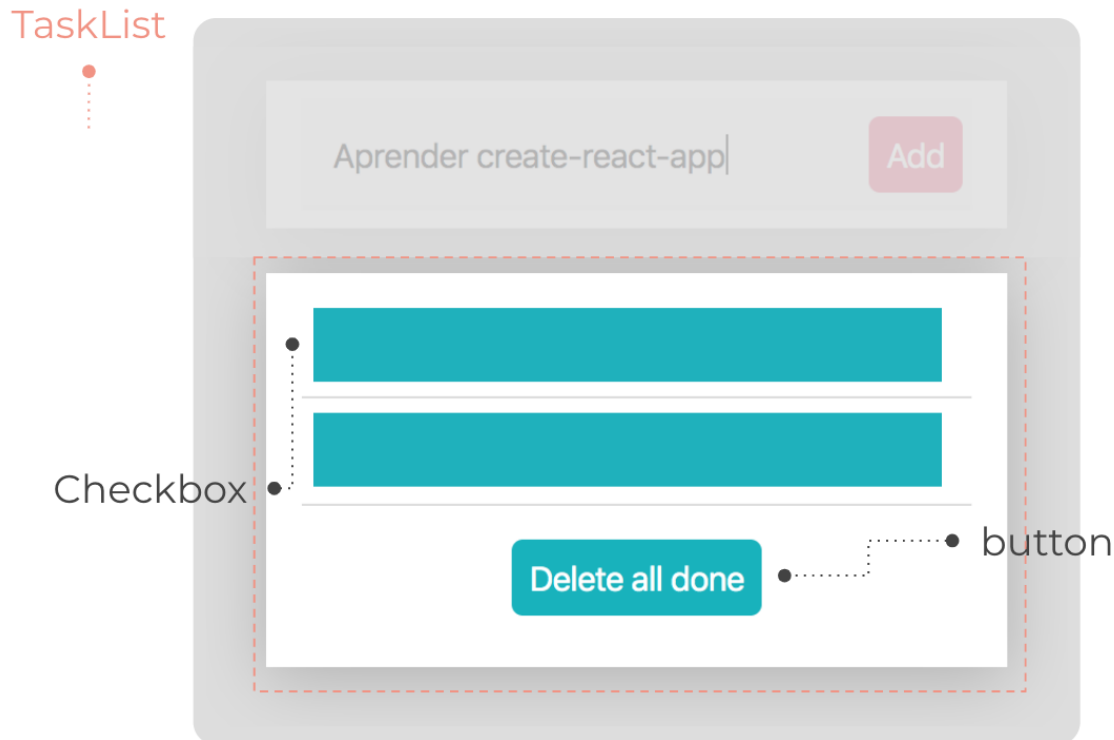
[Open in app](#)[Get started](#)

react-app-todo-Container-3.jsx hosted with ❤ by GitHub

[view raw](#)

C — TaskList

Vamos a darle forma al **componente funcional** `TaskList`, para que quede de la siguiente manera:



:: Vamos paso a paso

- Vamos a **recibir** de props la **propiedad** `list` y el **método** `setList`, la idea es **recorrer** `list` con `map`, para **generar** “*n*” **componentes** de `Checkbox`
- Si la `list` viene vacía, **mostramos el mensaje** `No task` y **ocultamos el botón**, caso **contrario**, **mostramos la lista** y el **botón**.

```
1 // ...
2 const TaskList = props => {
3   const { list, setList } = props;
```




[Open in app](#)
[Get started](#)

```

9      <div className="todo-list">
10        {list.length ? chk : "No tasks"}
11        {list.length ? (
12          <p>
13            <button className="button blue">
14              Delete all done
15            </button>
16          </p>
17        ) : null}
18      </div>
19    );
20  };
21  // ...

```

TaskList-1-1.jsx hosted with ❤ by GitHub

[view raw](#)

- Vamos a crear un **manejador de eventos** para `onChange` para el componente `checkbox`
- Vamos a **crear un método** `onChangeStatus`, para el **manejador** `onChange`
- El **método** `onChangeStatus` va a **recibir** los **datos** del `checkbox` que haya **cambiado**, por lo que debemos de **recorrer** con `map` nuestra **lista** `list`, y **actualizar** los **valores**.

```

1  // ...
2  const TaskList = props => {
3    const { list, setList } = props;
4
5    const onChangeStatus = e => {
6      const { name, checked } = e.target;
7      const updateList = list.map(item => ({
8        ...item,
9        done: item.id === name ? checked : item.done
10      }));
11      setList(updateList);
12    };
13
14    const chk = list.map(item => (
15      <Checkbox key={item.id} data={item} onChange={onChangeStatus} />

```



[Open in app](#)[Get started](#)

- Vamos a crear un **manejador de eventos** para `onClick` para el elemento `button`
- Vamos a **crear un método** `onClickRemoveItem`, para el **manejador** `onClick`
- El **método** `onClickRemoveItem`, vamos **filtrar la lista eliminando** todos los **ítems** que `done === true` y posteriormente **actualizar la lista** con `setList`

```
1 // ...
2 const TaskList = props => {
3   // ...
4   const onClickRemoveItem = e => {
5     const updateList = list.filter(item => !item.done);
6     setList(updateList);
7   };
8   // ...
9   return (
10     //...
11     <button className="button blue" onClick={onClickRemoveItem}>
12       Delete all done
13     </button>
14     // ...
15   );
16 };
17 // ...
```

TaskList-1-3.jsx hosted with ♥ by GitHub

[view raw](#)

:: Código completo

```
1 import React from "react";
2 import Checkbox from "../Checkbox";
3
4 const TaskList = props => {
5   // (C)
6   const { list, setList } = props;
7
```



[Open in app](#)[Get started](#)

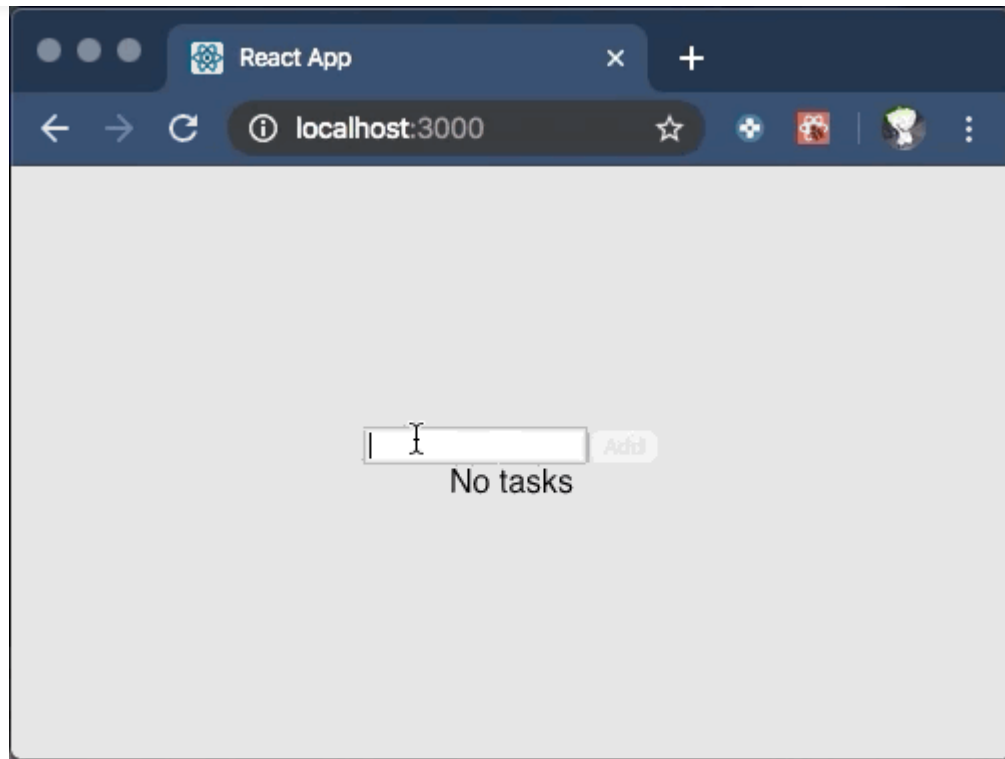
```
13     ...item,
14     done: item.id === name ? checked : item.done
15   }));
16   setList(updateList);
17 };
18
19 // (D)
20 const onClickRemoveItem = e => {
21   const updateList = list.filter(item => !item.done);
22   setList(updateList);
23 };
24
25 // (A-2)
26 const chk = list.map(item => (
27   <Checkbox key={item.id} data={item} onChange={onChangeStatus} />
28 ));
29 return (
30   <div className="todo-list">
31     {/*(A-1)*/}
32     {list.length ? chk : "No tasks"}
33     {/*(B)*/}
34     {list.length ? (
35       <p>
36         <button className="button blue" onClick={onClickRemoveItem}>
37           Delete all done
38         </button>
39       </p>
40     ) : null}
41   </div>
42 );
43 };
44
45 export default TaskList;
```

react-app-todo-Tasklist-2.jsx hosted with ♥ by GitHub

[view raw](#)

:: Ejemplo de la app

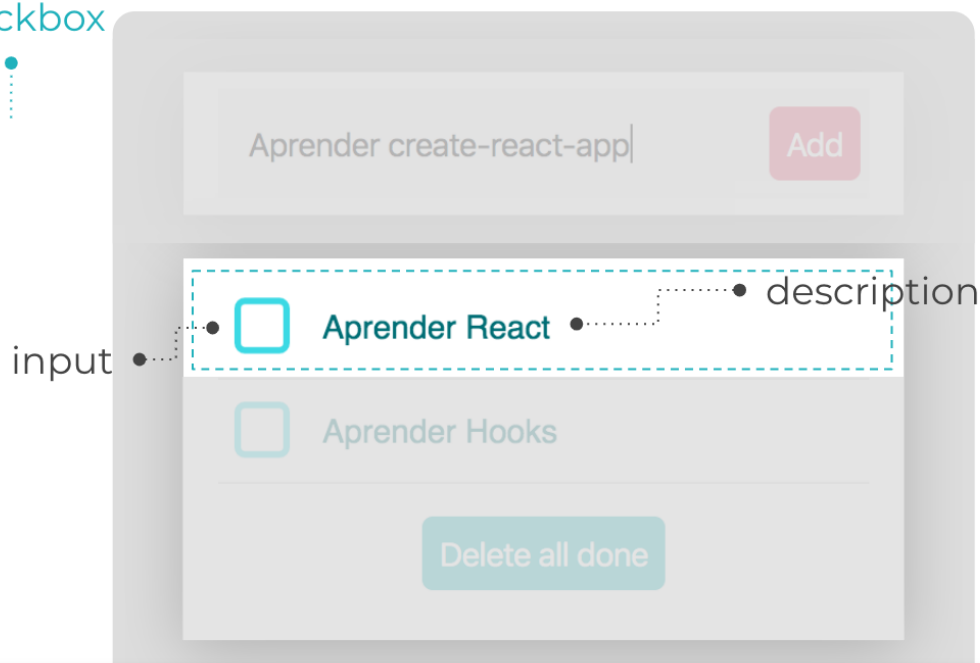


[Open in app](#)[Get started](#)

D— Checkbox

Vamos a darle forma al **componente funcional** checkbox , para que quede de la siguiente manera:

Checkbox



[Open in app](#)[Get started](#)

- Vamos a **recibir** de props la **propiedad** data y el **método** onChange .
- De data vamos a **abstraer** las **deseestructurar** id, description, done , **asignamos** las **propiedades** al JSX.
- En el **atributo** onChange del **elemento** input , vamos a **agregar** el **método** onChange , cuando se le de **click**, va a **ejecutar** el **método**.

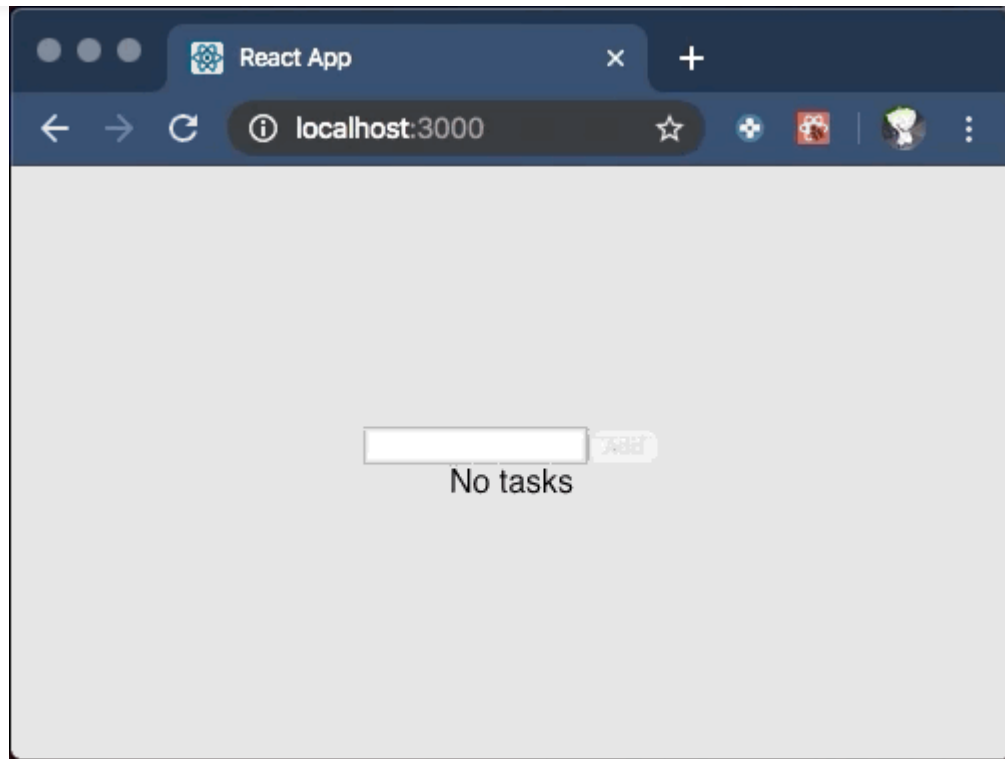
```
1  import React, { Fragment } from "react";
2
3  const Checkbox = props => {
4    // (A)
5    const {
6      onChange,
7      data: { id, description, done }
8    } = props;
9    return (
10     <Fragment>
11       <label className="todo new-item">
12         /*(B) (C)*/
13         <input
14           className="todo__state"
15           name={id}
16           type="checkbox"
17           defaultChecked={done}
18           onChange={onChange}
19         />
20       <div className="todo__text">{description}</div>
21     </label>
22   </Fragment>
23 );
24 };
25
26 export default Checkbox;
```

react-app-todo-Checkbox-2.jsx hosted with ❤ by GitHub

[view raw](#)

:: Ejemplo de la app



[Open in app](#)[Get started](#)

Hasta aquí ya tenemos al 100% la funcionalidad con React, ahora solo falta :

- **Agregar estilos**
- **Modificaciones al componente** Checkbox , ya que usa SVG .

Por ahora vamos **agregar** todos los **estilos** en `App.scss` , entonces, *copiar y pegar*:

```
1  html {  
2    background: #ddd;  
3    height: 100%;  
4    display: flex;  
5  }  
6  body {  
7    width: 100%;  
8    margin: auto;  
9  }  
10  
11  .App {  
12    font-family: sans-serif;
```



[Open in app](#)[Get started](#)

```
18   font-size: 15px;
19   max-width: 300px;
20   margin: auto;
21   margin-bottom: 20px;
22   padding: 8px 16px;
23   box-shadow: 0 5px 30px rgba(0, 0, 0, 0.2);
24 }
25
26
27 /** BUTTONS*/
28 $colorPink: #e95678;
29 $colorPinkHover: #c23455;
30
31 $colorBlue: #25b0bc;
32 $colorBlueHover: #166b72;
33
34 @mixin button($class, $color, $colorHover) {
35   .button.#{ $class } { background: $color; }
36   .button.#{ $class } :hover { background: $colorHover; }
37   .button.#{ $class } :active { background: $color; }
38 }
39
40 .button {
41   outline: none;
42   padding: 8px;
43   font-size: 15px;
44   border: 0px solid;
45   border-radius: 5px;
46   color: white;
47 }
48 .button[disabled] {
49   opacity: 0.3;
50 }
51
52 @include button('pink', $colorPink, $colorPinkHover);
53 @include button('blue', $colorBlue, $colorBlueHover);
54
55
56
```



[Open in app](#)[Get started](#)

```
62
63 .file-input > .text {
64   outline: none;
65   padding: 0px 10px;
66   font-size: 15px;
67   width: 230px;
68   height: 46px;
69   background: transparent;
70   border: 0px solid;
71 }
72
73
74 /** CHECKBOX*/
75 /* Original styles by Shaw : https://codepen.io/shshaw/pen/WXMdWE */
76 $duration: 0.4s;
77 $stroke: #42d7e4;
78 $colorText: #076b74;
79 $colorTextChecked: #a5e7ec;
80
81 .todo {
82   display: block;
83   position: relative;
84   padding: 1em 1em 1em 16%;
85   margin: 0 auto;
86   cursor: pointer;
87   border-bottom: solid 1px #ddd;
88   &:last-child { border-bottom: none; }
89 }
90
91 .todo.new-item {
92   opacity: 0;
93   transform: translateX(100px);
94   animation: fadeIn .3s linear forwards;
95 }
96
97 .todo__state {
98   position: absolute;
99   top: 0;
100  left: 0;
101  opacity: 0;
```



[Open in app](#)[Get started](#)

```
107   transition: all $duration/2 linear $duration/2;
108 }
109
110 .todo__icon {
111   position: absolute;
112   top: 0;
113   bottom: 0;
114   left: 0;
115   width: 100%;
116   height: auto;
117   margin: auto;
118
119   fill: none;
120   stroke: $stroke;
121   stroke-width: 2;
122   stroke-linejoin: round;
123   stroke-linecap: round;
124 }
125
126
127 .todo__line,
128 .todo__box,
129 .todo__check {
130   transition: stroke-dashoffset $duration cubic-bezier(.9,.0,.5,1);
131 }
132
133
134 .todo__circle {
135   stroke: $stroke;
136   stroke-dasharray: 1 6;
137   stroke-width: 0;
138
139   transform-origin: 13.5px 12.5px;
140   transform: scale(0.4) rotate(0deg);
141   animation: none $duration linear; //cubic-bezier(.08,.56,.04,.98);
142
143   @keyframes explode {
144     //0% { stroke-width: 0; transform: scale(0.5) rotate(0deg); }
145     30% {
```





Open in app

Get started

```

151     100% {
152         stroke-width: 0;
153         stroke-opacity: 0;
154         transform: scale(1.1) rotate(60deg);
155         //animation-timing-function: cubic-bezier(.08,.56,.04,.98);
156     }
157 }
158 }
159
160 .todo__box {
161     stroke-dasharray: 56.1053, 56.1053; stroke-dashoffset: 0;
162     transition-delay: $duration * 0.2;
163 }
164 .todo__check {
165     stroke: $stroke;
166     stroke-dasharray: 9.8995, 9.8995; stroke-dashoffset: 9.8995;
167     transition-duration: $duration * 0.4;
168 }
169 .todo__line {
170     stroke-dasharray: 168, 1684;
171     stroke-dashoffset: 168;
172 }
173 .todo__circle {
174     animation-delay: $duration * 0.7;
175     animation-duration: $duration * 0.7;
176 }
177
178 .todo__state:checked {
179     ~ .todo__text { transition-delay: 0s; color: $colorTextChecked; opacity: 0.6; }
180     ~ .todo__icon .todo__box { stroke-dashoffset: 56.1053; transition-delay: 0s; }
181     ~ .todo__icon .todo__line { stroke-dashoffset: -8; }
182     ~ .todo__icon .todo__check { stroke-dashoffset: 0; transition-delay: $duration * 0.6; }
183     ~ .todo__icon .todo__circle { animation-name: explode; }
184 }
185
186 @keyframes fadeIn {
187     to {
188         transform: translateX(0px);
189         opacity: 1;
190     }

```

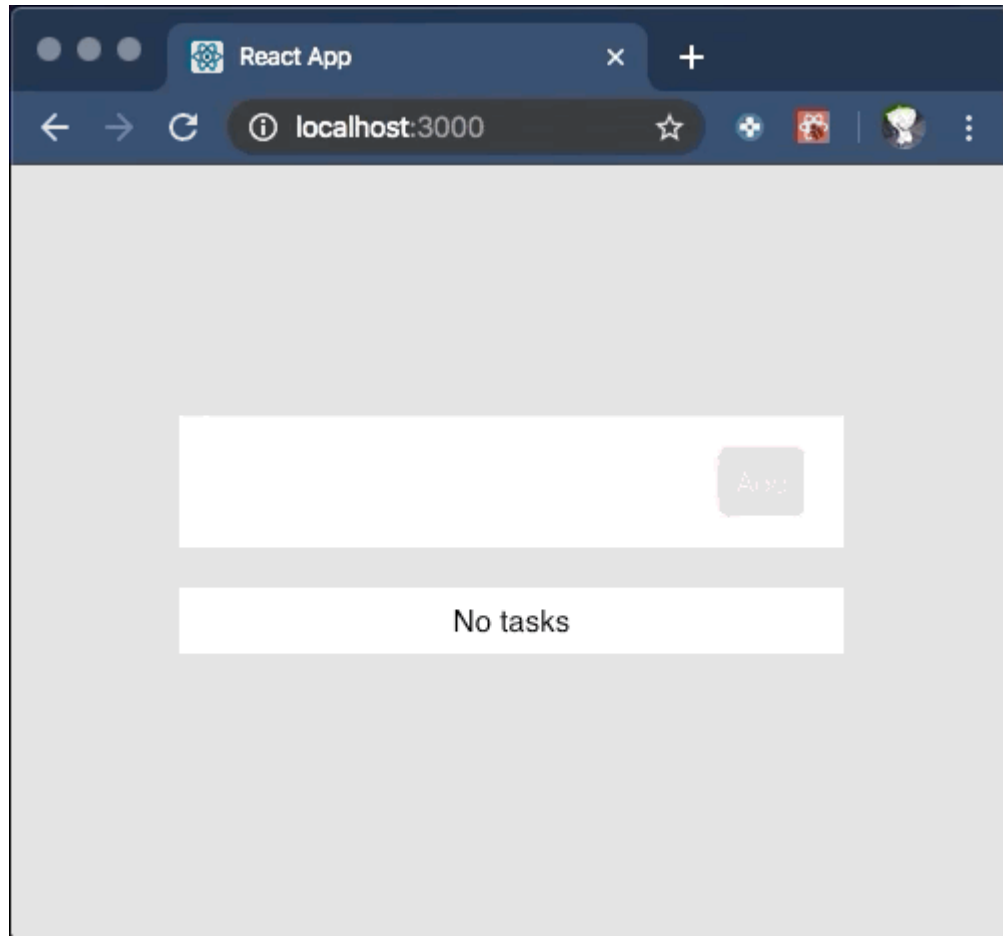


[Open in app](#)[Get started](#)

react-app-todo-App.scss hosted with ❤ by GitHub

[view raw](#)

:: Ejemplo de la app

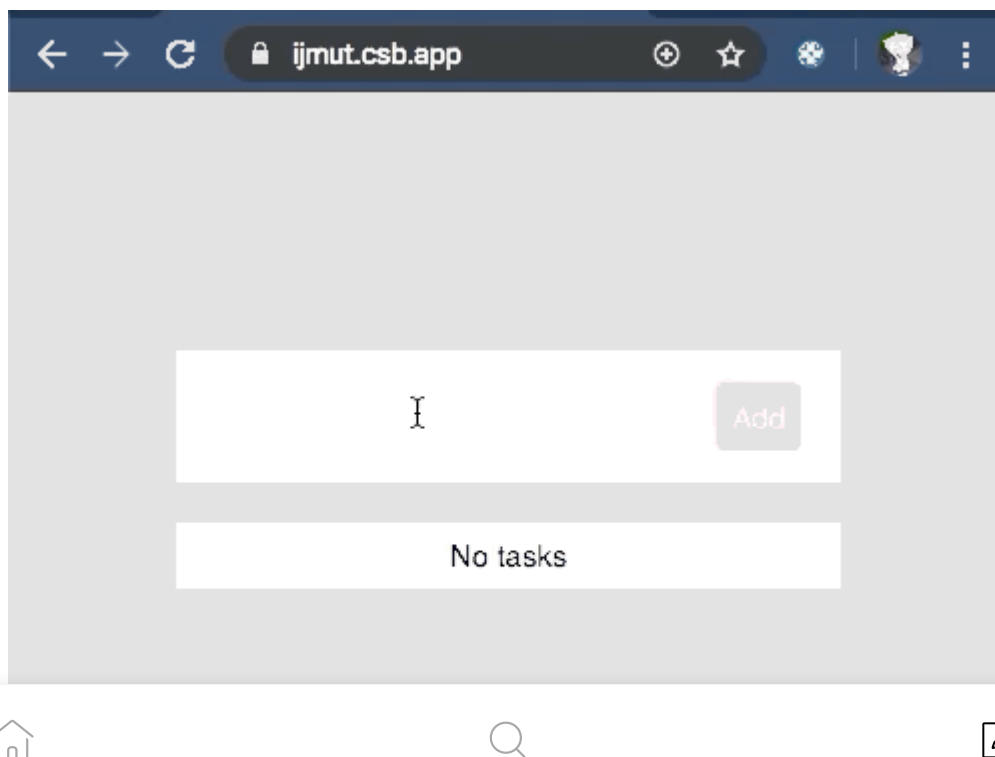


Para que se vea **animado** el checkbox , he sacado el **código** de **codepen**[\[ref\]](#), ahora **agregamos** el **svg** al **componente funcional** Checkbox :



[Open in app](#)[Get started](#)

:: Ejemplo de la app



[Open in app](#)[Get started](#)

:: Ejemplo completo de la app

:: Descarga ejemplo completo

- Ver código completo en [GitHub\[ref\]](#).

v. Construyendo la aplicación para subir a producción

Una vez que tenemos nuestro proyecto, el siguiente paso es **construir** el **proyecto** para **producción**, entonces, en **consola** debemos **ejecutar** el siguiente **comando** (*no olvides estar en la carpeta del proyecto*)



[Open in app](#)[Get started](#)

Si lo hemos hecho bien, la consola debe verse algo así:

```
Documents/app/my-todo-list-app ((master)) -[There are changes]-[There are changes]
→ yarn build
yarn run v1.22.4
$ react-scripts build
Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

 40.01 KB  build/static/js/2.0a7a9018.chunk.js
  1.34 KB  build/static/js/main.090dcb67.chunk.js
  1.17 KB  build/static/css/main.77305d17.chunk.css
   782 B   build/static/js/runtime-main.c6459e4e.js

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:

  yarn global add serve
  serve -s build

Find out more about deployment here:
```



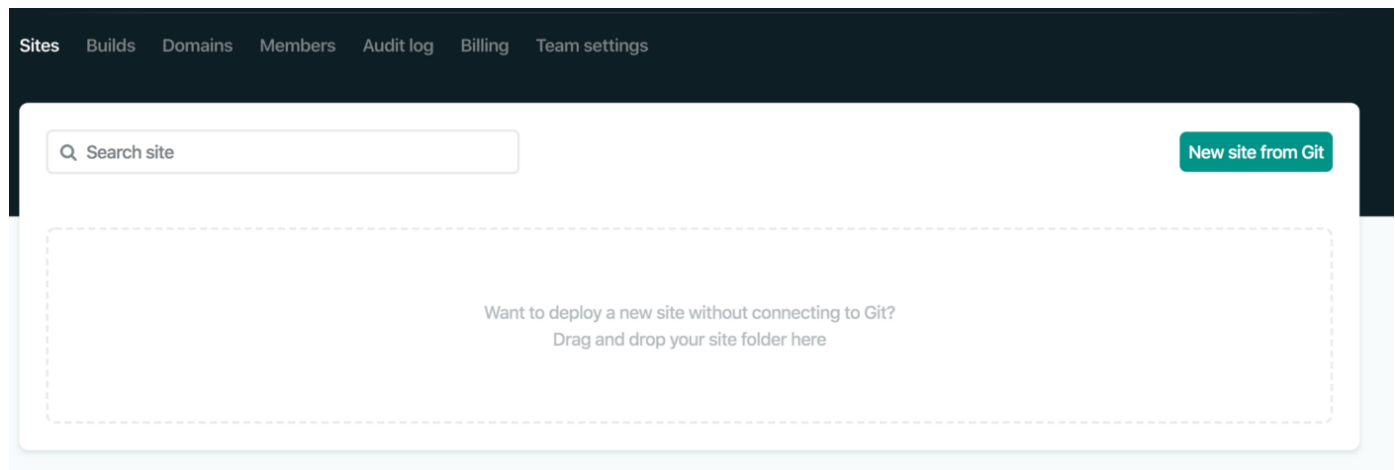
[Open in app](#)[Get started](#)

Con este **comando**, lo que hace es **crear** una **carpeta** dentro de la **aplicación**, llamada **build**, que es nuestro **proyecto** listo para mandar a **producción**.

vi. Subiendo el proyecto a un servidor (netlify)

Hay muchos servidores donde podemos alojar nuestra aplicación, para este **proyecto** vamos a utilizar **netlify**[\[ref\]](#).

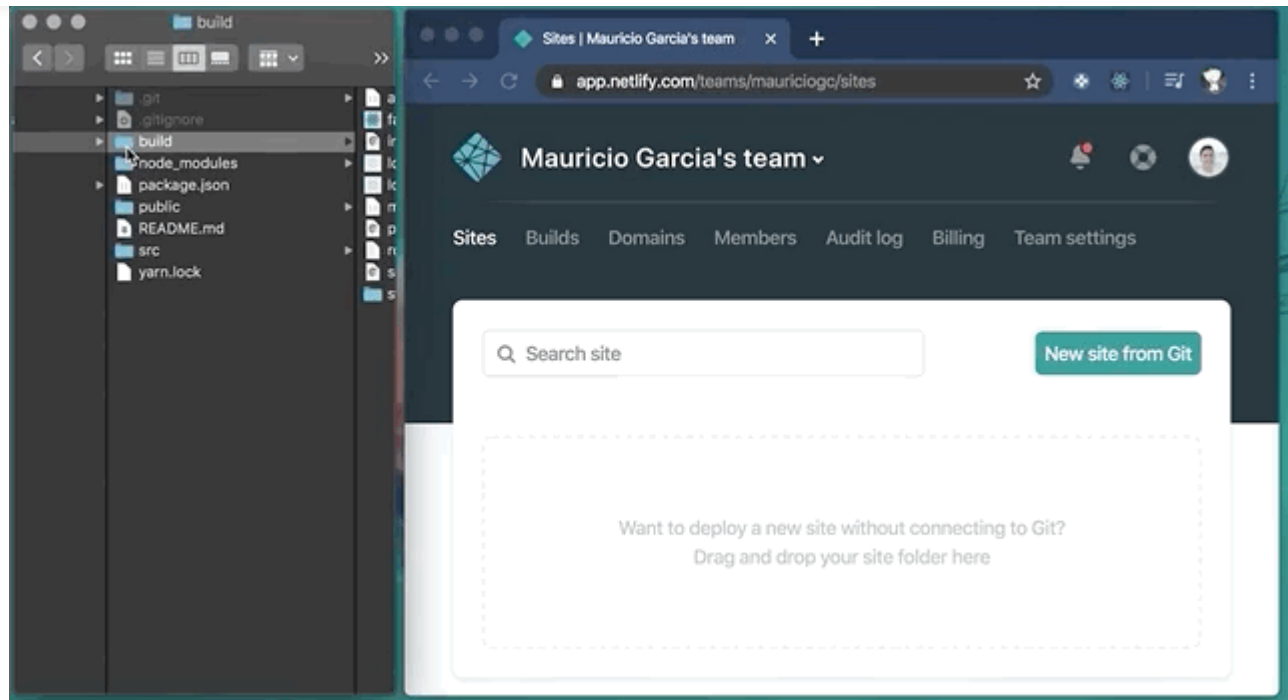
- Lo primero, es **entrar** a la **página** de **netlify**[\[ref\]](#)
- Nos vamos a **loguear** (es completamente **gratuito**)
- Una vez abierto, debemos de ver una pantalla como esta:



Tenemos varias formas de subir el proyecto:

- La **primera** y **más fácil**, es **arrastrar** la **carpeta** a la **página**, ver la siguiente imagen:

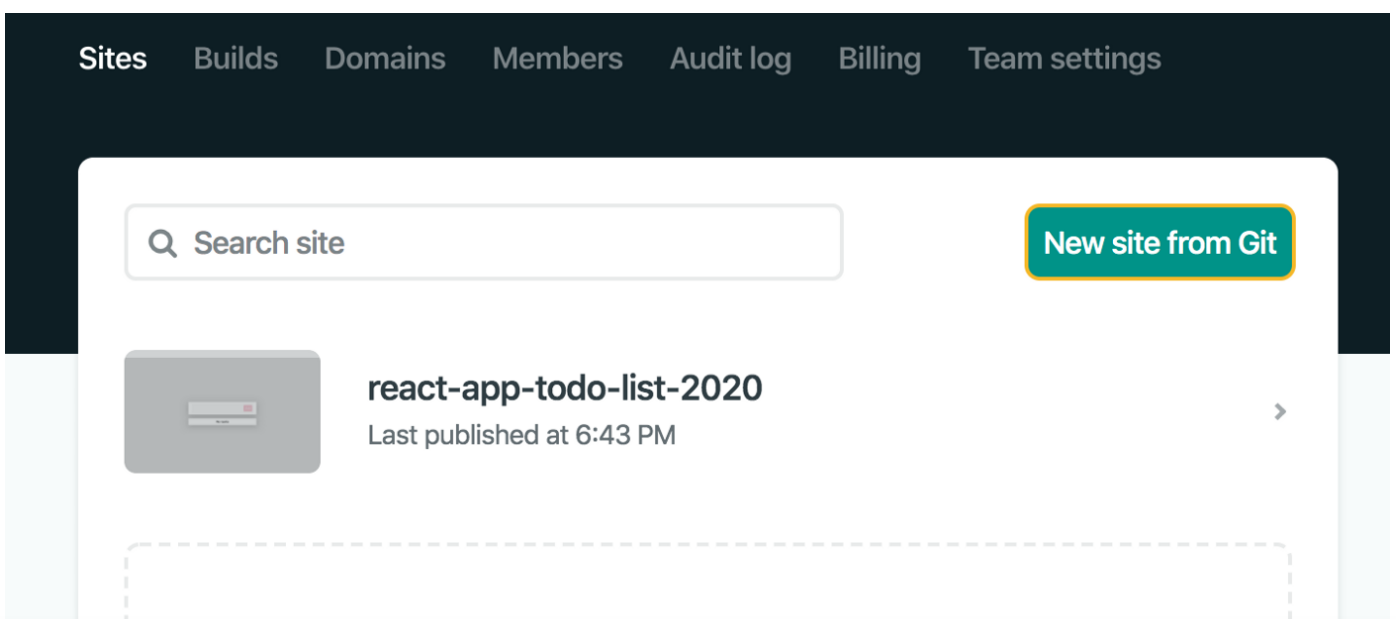


[Open in app](#)[Get started](#)

Y eso es todo!, te comparto la liga que he generado:

<https://react-app-todo-list-2020.netlify.com/>

- La otra forma es **subir** el **repositorio** a **Github**, una vez arriba, nos vamos al **dashboard**[\[ref\]](#), y le damos **click** al botón **New site from Git**



[Open in app](#)[Get started](#)

Create a new site

From zero to hero, three easy steps to get your site on Netlify.

1. Connect to Git provider

2. Pick a repository


3. Build options, and deploy!

Continuous Deployment

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.

 GitHub

 GitLab

 Bitbucket

Una vez que vinculamos nuestro usuario con la página, debe salir una pantalla como la siguiente:

Continuous Deployment: GitHub App

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.



mauriciogc ▾

 Search



mauriciogc/acamica



mauriciogc/animatable



mauriciogc/animate.css



[Open in app](#)[Get started](#)

Buscamos el **proyecto** que vamos a **subir**, y le **damos click**, después le **decimos** de que **rama** va a tomar el **proyecto** y que **comando** es el que va a **utilizar** para **construirlo**, así como la **carpeta** que va a **subir** y por **último** le damos **click** a **Deploy site**

Deploy settings for mauriciogc/react-app-todo-list

Get more control over how Netlify builds and deploys your site with these settings.

Owner

Mauricio Garcia's team



Branch to deploy

master



Basic build settings

If you're using a static site generator or build tool, we'll need these settings to build your site. [Learn more in the docs](#) ↗

Build command

npm run build



Publish directory

build/



Show advanced

Deploy site

Cuando haya **terminado**, nos debe salir algo como la siguiente imagen:



[Open in app](#)[Get started](#)

brave-colden-b7845b

- <https://brave-colden-b7845b.netlify.com>

Manual deploys. Last published at 6:43 PM.

[⚙ Site settings](#)[⚙ Domain settings](#)

*Nota: La **desventaja** de hacerlo así es que **toma más tiempo construirlo**, que subirlo directamente la carpeta, pero la **ventaja** es de que **siempre** va a estar **vinculado al proyecto**.*

En la siguiente entrega vamos a ver **React — Valores por defecto con defaultProps y valores predeterminados con ES6**

La entrega pasada vimos **React — Mi primera App con create-react-app**

Bibliografía y links que te puede interesar...

mauriciogc/my-todo-list-app

See GIF example --- Coming soon --- In the project directory, you can run: Runs the app in the development mode. Open...

[github.com](#)

facebook/create-react-app



[Open in app](#)[Get started](#)

A Beginner's Guide to Sass

What is Sass?

levelup.gitconnected.com

React App

Web site created using create-react-app

react-app-todo-list-2020.netlify.com

