La estructura de directorios de cada proyecto de React puede ser diferente. Para establecer un punto de comienzo común, Facebook creó create-react-app. Este proyecto genera la estructura básica de una aplicación desarrollada con React.

El objetivo de este proyecto es establecer un punto simple y básico para comenzar a desarrollar nuestra aplicación. Nosotros vamos a partir de esta aplicación, pero introduciremos algunas variaciones que nos ayudarán a tener un código más organizado y claro.

Creando nuestra aplicación

Requisitos

Para poder ejecutar `create-react-app` necesitamos disponer de nodejs. **Este requisito solo es necesario en desarrollo**. Una vez que tengamos los ficheros compilados ya no lo necesitamos.

La versión mínima requerida de `node` es la 4. No obstante, para mejorar la velocidad de ejecución recomiendan instalar como mínimo la versión 6.

Podéis instalar `node` directamente desde su web. Yo os recomiendo utilizar el gestor de versiones de `node` llamado NVM. Con este gestor podéis cambiar la versión actual de `node` con un solo comando. Es muy útil cuando para trabajar con distintos proyectos que requieren entornos diferentes.

Instalar create-react-app

Esta aplicación de consola (*CLI*) se distribuye como paquete npm. Para los que no estéis familiarizados con `node` y su entorno, `npm` es un gestor de paquetes. Su funcionamiento es muy similar a *Bundler* para ruby o *Composer* para PHP.

Para instalar esta aplicación ejecutamos el siguiente comando en nuestro terminal.

```
npm install -g create-react-app
```

La opción `-g` instala este paquete de manera global. Así tendremos el comando `create-react-app` disponible en cualquier directorio en la consola. A continuación, creamos nuestra aplicación con esta CLI.

```
create-react-app github-releases
```

Este comando genera una nueva carpeta llamada `github-releases` con todo lo necesario para empezar a desarrollar nuestro proyecto. También se encarga de instalar todas las dependencias.

Vamos a iniciar la aplicación para comprobar que todo ha ido bien.

```
cd github-releases && npm start
```

Si la instalación ha ido bien, se abrirá nuestro navegador con la url http://localhost:3000. Por defecto, `create-react-app` incluye la siguiente página de inicio.

Estructura de directorios

`create-react-app` genera la siguiente estructura de directorios.

```
github-releases/
  README.md
  node_modules/
  package.json
  .gitignore
  public/
  src/
```

- `README.md` : contiene documentación adicional sobre `create-react-app`
- `node_modules` : carpeta de `node` que contiene todas las dependencias instaladas
- `package.json` : fichero que define nuestro proyecto, su configuración y sus dependencias
- `public` : carpeta pública. Todo lo incluido aquí se servirá como ficheros estáticos
- `src` : contiene el código fuente de nuestra aplicación

Vamos a entrar más en detalle en los ficheros que contiene la carpeta `src` .

```
github-releases/
  src/
    App.css
    App.js
    App.test.js
    index.css
    index.js
```

```
index.js
```

- `App.*` : estos ficheros contienen los estilos, el código fuente y los tests del componente `App`
- `index.*` : son los ficheros principales de nuestra aplicación, el punto de entrada. En estos ficheros se definen los estilos básicos y se monta la aplicación (*ReactDOM*) en el DOM.

Cómo vemos la estructura es muy sencilla. Sin embargo, pensemos ahora en una aplicación más compleja. Cualquier aplicación por pequeña que sea contendrá como mínimo varios componentes. **Si mantenemos esta estructura tendremos en la misma carpeta una gran cantidad de ficheros sin ningún tipo de organización** más que sus nombres.

Por este motivo vamos a modificar esta estructura para organizar mejor nuestros componentes. Creamos dos nuevas carpetas dentro de `src` :

- `src/components` : esta carpeta contendrá los distintos componentes de tipo *presential* de nuestra aplicación

- `src/containers` : como su nombre indica, esta contendrá los componentes de tipo *container*

Con este cambio la estructura de la carpeta `src` de nuestro proyecto es la siguiente:

```
github-releases/
  src/
    components/
      App.css
      App.js
      App.test.js
      logo.svg
    containers/
    index.css
    index.js
```

El único cambio que necesitamos hacer en el código es actualizar la ruta de importación del componente `App` en `src/index.js` :

```
import React from 'react';
import ReactDOM from 'react-dom';
- import App from './App';
+ import App from './components/App';
```

Aún podemos mejorar esta estructura. Del mismo modo que `src` , la carpeta `components` **también puede contener una gran cantidad de ficheros**. Además, como ocurre con `App` , los componentes pueden requerir otro tipo de ficheros (`svg` , `json` …). Si queremos exportar un componente a otro proyecto necesitaremos comprobar su código fuente para conocer todas sus dependencias.

Para solventar este problema podemos englobar todos los ficheros requeridos de un componente en una única carpeta. De este modo, exportar un componente será una tarea mucho más sencilla. Además, tendremos nuestros componentes mejor organizados.

```
github-releases/
  src/
    components/
      App
        App.css
        App.js
        App.test.js
        index.js
        logo.svg
    containers/
    index.css
    index.js
```

Si os fijáis, tenemos un nuevo fichero `index.js` dentro de `src/components/App`. Su contenido es:

```
import App from './App';
export default App;
```

Este fichero nos facilita la labor de importar el componente. Si no lo existiese, tendríamos que actualizar la ruta del componente `App` a `src/components/App/App` en `src/index.js`. Lo que resulta muy redundante. Esto es posible gracias a que cuando importamos una carpeta en JavaScript, `node` busca el fichero `index.js` por defecto.

Detrás de create-react-app

`create-react-app` es una gran aplicación para comenzar un proyecto de React de una manera muy rápida y sencilla. Con dos comandos tenemos nuestra aplicación disponible en el navegador. En otras palabras, podemos enfocarnos en escribir código.

Aún así, merece la pena pararnos a ver las librerías más importantes que incluye `create-react-app`:

- Babel: cómo cabría esperar, necesitamos las nuevas funcionalidades de ES6
- Webpack: esta librería compila todas las dependencias de nuestro proyecto y genera los ficheros estáticos. Entraremos en más detalle en el siguiente apartado.
- webpack-dev-server: es un servidor de desarrollo que sirve nuestros componentes e incluye algunas funcionalidades avanzadas como `auto-reload`
- ESLint: comprueba nuestro código en busca de fallos de sintaxis y malas prácticas

- Jest: framework de testing para Javascript creado por Facebook

Modificar la configuración

Este tipo de aplicaciones son muy útiles pero no debemos olvidar que no eliminan la complejidad sino que la ocultan. En la mayoría de los casos esto no es problema, pero si necesitamos modificar la configuración empiezan los dolores de cabeza. Este es uno de los aspectos que más valoro de esta aplicación: los desarrolladores que hay detrás han pensando en esta situación.

En nuestro caso, no necesitaremos llegar a modificar la configuración. Pero si fuera necesario, el comando `npm eject` copia todos los ficheros de configuración al proyecto y elimina `create-react-app` . **Esto nos proporciona el control que necesitamos**.

React dev tools

Esta extensión de Google Chrome nos permite visualizar el árbol de componentes de nuestra aplicación así como las propiedades y el estado de cada uno de ellos. También podemos modificar el estado de cada componente y ejecutar métodos de estos.

Para descargarla, sólo tenéis que dirigiros a su página en Chrome Extensions store.