Cómo crear un plugin de WordPress con React

Mauricio Bruno. WordPress es una gran plataforma que apoya varias páginas y componentes. Y justo por ser tan grande, puede intimidar a quienes quieren desarrollarse allí, ¡incluso más cuando abrimos la documentación! Pero no te preocupes. Este artículo muestra que crear tus primeros plugins y blocks en WordPress es relativamente fácil. Te mostraré cómo crear un plugin con el mínimo de códigos, así como integrar una app de React en él. ¿Lo descubrimos ahora?

Motivación

Digamos que ya tienes un sitio web hecho en WordPress, pero te pidieron añadirle un detalle, por ejemplo, agendar citas médicas en una plataforma específica. En este caso, difícilmente encontrarás un plugin ya hecho con la funcionalidad deseada. Tampoco es viable que reconstruyas todo el Front end para agregar esta función. Por lo tanto, una buena alternativa es crear un plugin con React.

Solo crearemos parte de esa función acá porque mi meta es que sea tan simple como sea posible. Sin embargo, lo que haremos acá te permitirá insertar una aplicación de React en cualquier página y, desde ahí, crear lo que desees. ¡Vamos!

Crear el plugin

Asumiendo que ya tienes instalado el entorno de desarrollo de WordPress, ve a la carpeta **wp-content/plugins/** y crea una nueva para el plugin:

mkdir react-plugin

cd react-plugin

El primer paso es crear el archivo principal, que en nuestro caso será **react-plugin.php**:

<?php

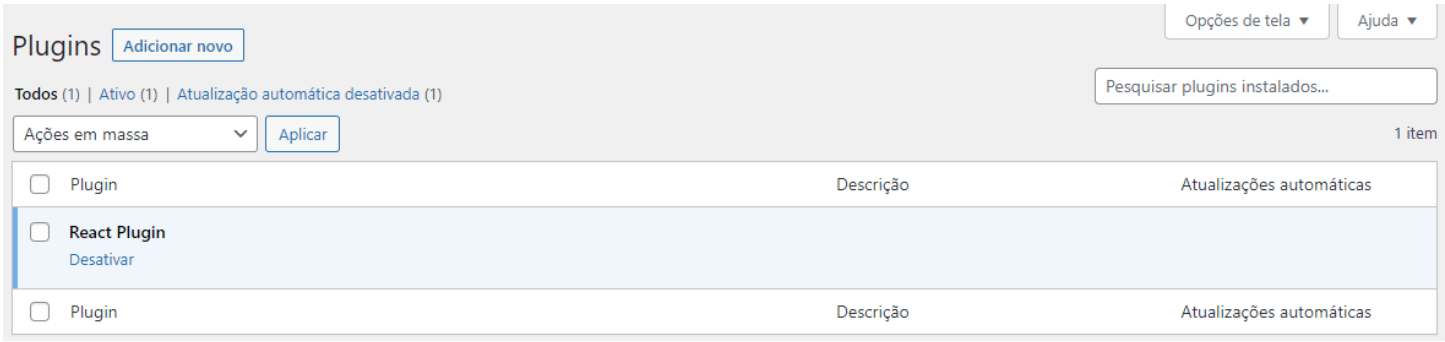
*/**

** Plugin Name: React Plugin*

**/*

Dejaremos esta configuración mínima para asegurar que funcione el plugin. Puedes ver todos los campos disponibles para configurar el plugin aquí.

Ahora, al ir al tablero de WordPress, incluso sin ninguna función, podrás ver que el plugin ya está disponible para activarse, por lo que puedes iniciarlo sin problemas.



A continuación, ¡comencemos el proyecto con React! Para lograrlo, iniciemos npm e instalemos la dependencia **@wordpress/scripts**.

npm init -y

npm install --save-dev @wordpress/scripts

El paquete @wordpress/scripts agrega varios detalles como React, ReactDOM, Babel, Sass yTypescript, e incluso nos ayuda a construir nuestro código. Solo con él, podemos hacer muchísimo.

Ahora necesitamos configurar tres scripts en el archivo **package.json** porque los necesitaremos para desarrollar el plugin:

{

...

"scripts: {

"build": "wp-scripts build

"start": "wp-scripts start",

"plugin-zip": "wp-scripts plugin-zip"

},

}

Es momento de armar el punto de entrada para nuestra app de React. Creemos una carpeta src donde pondremos nuestro código. Dentro de ella estará un archivo i**ndex.jsx**:

import React from "react";

import ReactDOM from "react-dom";

import { App } from "./App";

const containers = document.querySelectorAll(".react-plugin");

containers.forEach((container) => {

ReactDOM.render(<App />, container);

});

El snippet **document.querySelectorAll(".react-plugin")** regresa una lista de todos los elementos en la página que tienen la clase react-plugin. Por cada elemento retornado, rendereamos nuestra app. De esta forma, nuestro plugin puede insertarse más de una vez en la misma página.

Creemos ahora el componente **src/App.jsx**:

import React, { useState } from "react";

import "./App.scss";

export function App() {

const [counter, setCounter] = useState(0);

const increment = () => setCounter(counter + 1);

return <button onClick={increment}>Cliked me {counter} times!</button>;

}

Y apliquemos algunos estilos con **src/App.scss**:

.react-plugin {

display: flex;

justify-content: center;

padding: 1.5rem;

background-color: black;

button {

border: 0;

border-radius: 0.5rem;

padding: 1rem 0.75rem;

cursor: pointer;

&:hover {

background-color: lightgray

}

}

}

OK, hemos creado un componente contador simple y usado algunos estilos para probar que puedes usarlos, pero ahora necesitamos mostrarlos en alguna parte. ¿Cómo lo hacemos?

Registrar los scripts

En una app de React regular, crearíamos una página HTML e incluiríamos allí el link de nuestro script. WordPress es similar en el hecho de que maneja las páginas y necesitamos decirle que incluya nuestros scripts y estilos. Para hacero, usamos algunas funciones que provee.

Finalicemos ahora el archivo **react-plugin.php**:

```
<?php

/**

* Plugin Name: React Plugin

*/

function react_plugin_shortcode()

{

wp_enqueue_script(

"react_plugin_js",

plugin_dir_url(__FILE__) . "/build/index.js",

[ "wp-element"],

"0.1.0",

true

);

wp_enqueue_style(

"react_plugin_css",

plugin_dir_url(__FILE__) . "/build/index.css"

);

return "<div class='react-plugin'></div>"
```

}

add_shortcode("react-plugin", "react_plugin_shortcode");

Aquí definimos la función **react_plugin_shortcode**. Dentro de esta función insertamos los scripts en la página mediante la función **wp_enqueue_script** y el CSS con **wp_enqueue_style**. El segundo argumento de estas funciones es el camino a los archivos que se insertarán. Yo lo colocaré donde estarán los archivos después de que construyamos el código.

Ten en cuenta que como tercer argumento es **wp_enqueue_script**, pasamos **wp-element package** como dependencia de nuestro script porque es desde este paquete que React vendrá cuando se ejecute el código en la página. Pero si **@wordpress/scripts** ya instalado ya provee React, entonces ¿por qué debemos colocar aquí esta dependencia?

Lo que sucede es que WordPress ya tiene React como núcleo, así que cuando construimos el código, **@wordpress/scripts** no incluirá el código React en nuestro plugin. Por lo tanto, le decimos a WordPress que nuestro script lo use a través de esta dependencia. Esto es bueno porque ayuda a reducir, al final, el tamaño del código generado. En pocas palabras, **@wordpress/scripts** nos ayuda en la etapa de desarrollo.

Todavía dentro de la función, regresamos el código que reemplazará el shortcode, el cual es el elemento contenedor de nuestra app.

Finalmente, enlazamos la función al shortcode del react-plugin, de mode que cada vez que WordPress encuentre el texto [react-plugin] en una página llame a nuestro proceso.

Ahora construyamos el código:

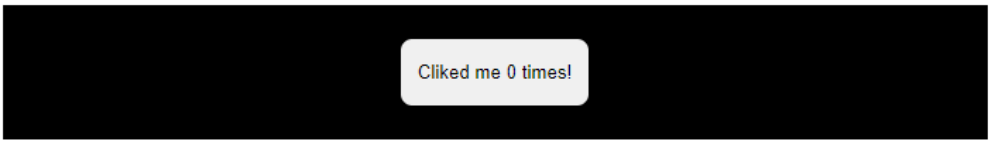
npm run build

Voilà!

¿Todo listo? Lo único que falta es editar cualquier página o post y colocar el shortcode que creamos, [react-plugin]:

Finalmente, veremos el componente React ya presente:

Hello world!



¿Ves lo simple que fue? Incluso podemos ir más allá y crear un block en lugar de usar un shortcode. Esto hace aún más fácil usar nuestro componente.

Crear un block

Hagamos algunos cambios pequeños a nuestro código. Lo primero será añadir el paquete **@wordpress/blocks** y usar el script de inicio para construirlo automáticamente:

npm install @wordpress/blocks

npm run start

Necesitaremos editar el archivo plugin principal (react-plugin.php). Nota que el código lucirá similar a lo que teníamos antes:

```
<?php

/**

* Plugin Name: React Plugin

*/

function react_plugin_block_init()

{

register_block_type(

__DIR__ . '/build',

['render_callback' => 'react_plugin_render_block']

);

}
```

```
function react_plugin_render_block()

{

wp_enqueue_script(

"react_plugin_frontend_js",

plugin_dir_url(__FILE__) . "/build/frontend.js",

["wp-element"],

"0.1.0",

true

);

return "<div class='react-plugin'></div>"

}

add_action('init', 'react_plugin_block_init');
```

Creamos aquí la función r**eact_plugin_block_init**, el cual registrará nuestro block usando la función que WordPress provee: **register_block_type**. Como primer argumento, pasamos la ruta a una carpeta, por lo que WordPress buscará un archivo **block.json** en esta carpeta, donde pondremos la configuración del block.

En el segundo argumento, pasamos otra función que creamos para ejecutar cuando el block esté rendereado en la página. En esta función, igual que antes, insertamos un script en la página y regresamos el elemento contenedor, pero esta vez, dividiremos el código en dos archivos: el que usamos, **index.js**; y el segundo, **frontend.js**.

Por último, llamamos **add_action** para enlazar nuestras funciones al init hook, lo cual hará que se ejecute nuestro código apenas se cargue WordPress.

Block.json

WordPress recomienda colocar nuestra configuración de block en un archivo **block.json** (consulta la documentación completa aquí). Creémosla ahora dentro de la carpeta src.

```
{

"$schema": "<https://schemas.wp.org/trunk/block.json>"

"apiVersion": 2,

"name": "react-plugin/counter",
```

"title": "React Plugin Counter",

"category": "layout",

"icon": "button",

"editorScript": "file:./index.js",

"viewScript": "file:./frontend.js",

"style": "file:./index.css"

}

Nota los tres últimos settings: **editorScript**, **viewScript**, y **style**. Allí definimos cuál código se usará cuando estemos en la pantalla de edición de la página, el código usado en el "front-end" y el CSS usado por ellos, respectivamente.

Vayamos ahora al código de JavaScript. Como dije antes, lo dividiremos en dos partes. Primero, **src/index.js**:

import { registerBlockType } from "@wordpress/blocks";

import { App } from "./App";

import metadata from "./block.json";

registerBlockType(metadata.name, {

edit: App,

});

En este archivo, registramos nuestro block en JavaScript, lo cual es necesario para que funcione. Pasamos nuestro componente de app a la opción de editar porque debemos pasar lo que se mostrará en el apartado de admin de la pantalla de edición de la página. Finalmente, el archivo **src/frontend.jsx**:

import React from "react";

import ReactDOM from "react-dom";

import { App } from "./App";

const containers = document.querySelectorAll(".react-plugin");

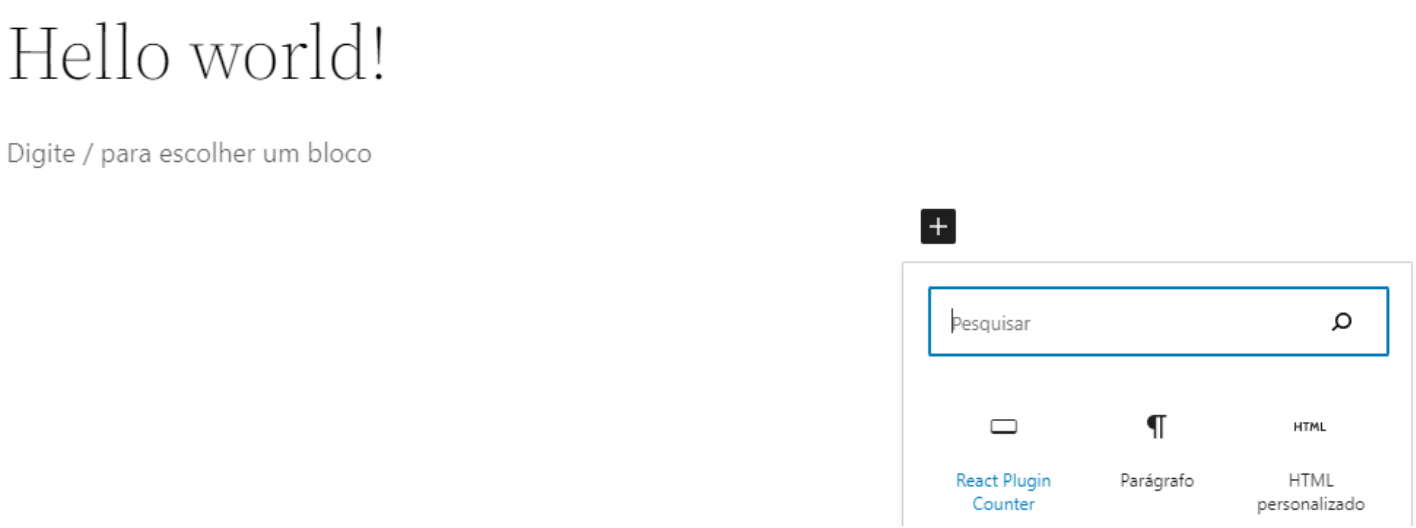
containers.forEach((container) => {

ReactDOM.render(<App />, container);

});

Aquí conservamos lo que estaba en nuestro archivo previo, de modo que este código será ejecutado solo cuando se muestre la página, causando que el block sea renderizado.

Con todo listo, solo falta agregar el block a la página.



¡Eso es todo! Cuando veas la página, el componente React debería mostrarse sin problemas.

¡Felicidades!

¿Viste cuán fácil es crear un plugin de WordPress plugin con React? Por supuesto, lo que hicimos fue algo simple, con configuración mínima, pero la meta era mostrarte cómo hacerlo.

Para terminar, construyamos nuestro código y generemos un archivo zip. Ejecutemos en la terminal los siguientes comandos:

npm run build

npm run plugin-zip.

¡Listo! ¡Ya tienes tu plugin listo para distribución! Supón que quieres conocer más sobre el tema. En ese caso, te recomiendo que revises la documentación oficial de WordPress, especialmente el paquete **[@wordpress/create-block]** a través de este enlace, el cual te ayudará mucho a crear blocks y otros elementos. Espero haberte ayudado.

⚠️