## Preparemos nuestra aplicación para convertirla en una PWA

​  
Lo primero que tenemos que crear es nuestro service-worker.js dentro de la carpeta /public:  
​

**const** doCache = **false**;

**const** CACHE\_NAME = 'pwa-cache';

​

**self**.addEventListener("activate", event => {

**const** cacheWhitelist = [CACHE\_NAME];

event.waitUntil(

caches.keys()

.then(keyList =>

Promise.all(keyList.map(key => {

**if** (!cacheWhitelist.includes(key)) {

console.log(`Deleting cache: ${key}`)

**return** caches.delete(key);

}

}))

)

);

});

​

**self**.addEventListener('install', **function** (event) {

**if** (doCache) {

event.waitUntil(

caches.open(CACHE\_NAME)

.then(**function** (cache) {

fetch("manifest.json")

.then(response => {

response.json()

})

.then(assets => {

**const** urlsToCache = [

"/",

assets["bundle.js"]

]

cache.addAll(urlsToCache)

console.log('cached');

})

})

);

}

});

​

**self**.addEventListener('fetch', **function** (event) {

**if** (doCache) {

event.respondWith(

caches.match(event.request).then(**function** (response) {

**return** response || fetch(event.request);

})

);

}

});

​  
La configuración previa nos permite registrar nuestra aplicación debidamente. Ahora vamos a crear el archivo de configuración donde podremos agregar el nombre de nuestra aplicación, ícono entre otras características importantes que describen la aplicación.  
​  
Crear el archivo manifest.json en la carpeta /public:

{

"short\_name": "Platzi Conf Store",

"name": "A simple Store",

"icons": [

{

"src": "assets/icon.png",

"sizes": "512x512",

"type": "image/png"

}

],

"start\_url": "/",

"background\_color": "#222",

"theme\_color": "#222",

"display": "standalone"

}

​  
Descarga el ícono de React de la sección de recursos y agrégalo a la carpeta /public.  
​

## Configurando Webpack para copiar los archivos necesarios al compilar el proyecto:

​  
Para continuar con el proceso de la construcción de la PWA es necesario instalar un plugin para webpack que nos ayudará a copiar los archivos de la carpeta /public a la carpeta de nuestro proyecto compilado.  
​

npm **install** copy-webpack-**plugin** --save-dev

​  
Una vez instalado, es necesario agregar la configuración necesaria en el archivo wepack.config.js  
​  
Primero importamos copy-webpack-plugin en la parte superior del archivo  
​

**const** CopyPlugin = require('copy-webpack-plugin');

​  
Agregamos la configuración necesaria en la sección de plugins:  
​

**new** **CopyPlugin**({

patterns: [

{ from: 'public/manifest.json', to: '' },

{ from: 'public/service-worker.js', to: '' },

{ from: 'public/icon.png', to: 'assets' },

],

}),

​  
En este caso estamos agregando 3 elementos a la carpeta dist, el primero es nuestro archivo manifest, luego agrega el archivo service-worker.js y al final el archivo icon.png, de esta forma al compilar el proyecto podemos disponer de estos archivos dentro del compilado final.  
​

## Agregar service worker a index.html

​  
Una de las tareas que tenemos que realizar antes de probar nuestra aplicación es agregar el soporte del service worker al proyecto creando la lógica siguiente dentro de index.html.  
​

<script>

**if** ('serviceWorker' **in** navigator) {

window.addEventListener('load', **function** () {

navigator.serviceWorker.register('service-worker.js').then(**function** (registration) {

console.log('ServiceWorker registration successful with scope: ', registration.scope);

}, **function** (err) {

console.log('ServiceWorker registration failed: ', err);

}).catch(**function** (err) {

console.log(err)

});

});

} **else** {

console.log('service worker is not supported');

}

</script>

​  
Ahora demos de agregar la referencia al archivo manifest.json dentro de la etiqueta <head>:  
​

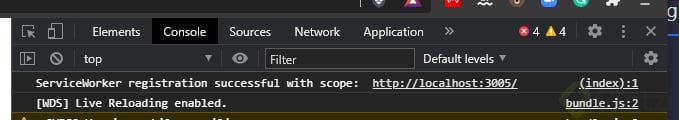
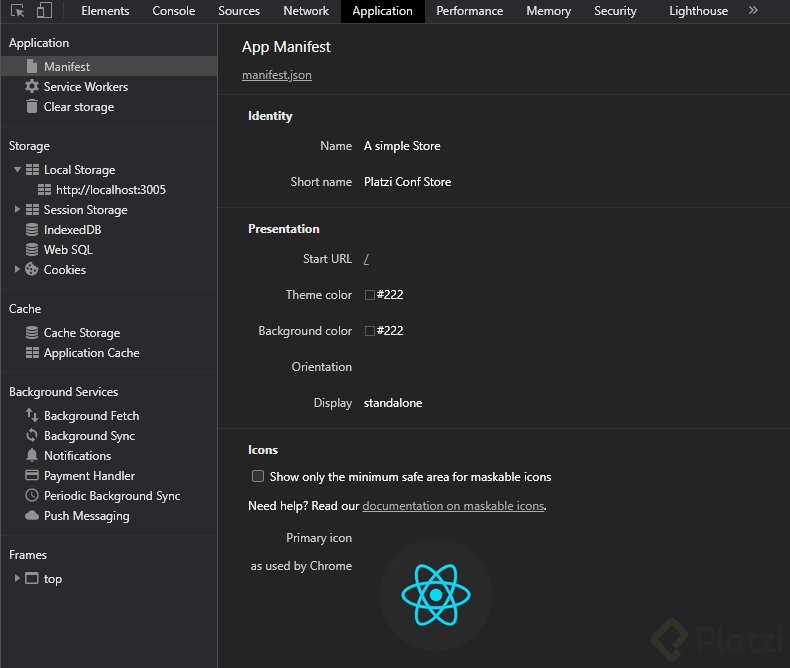
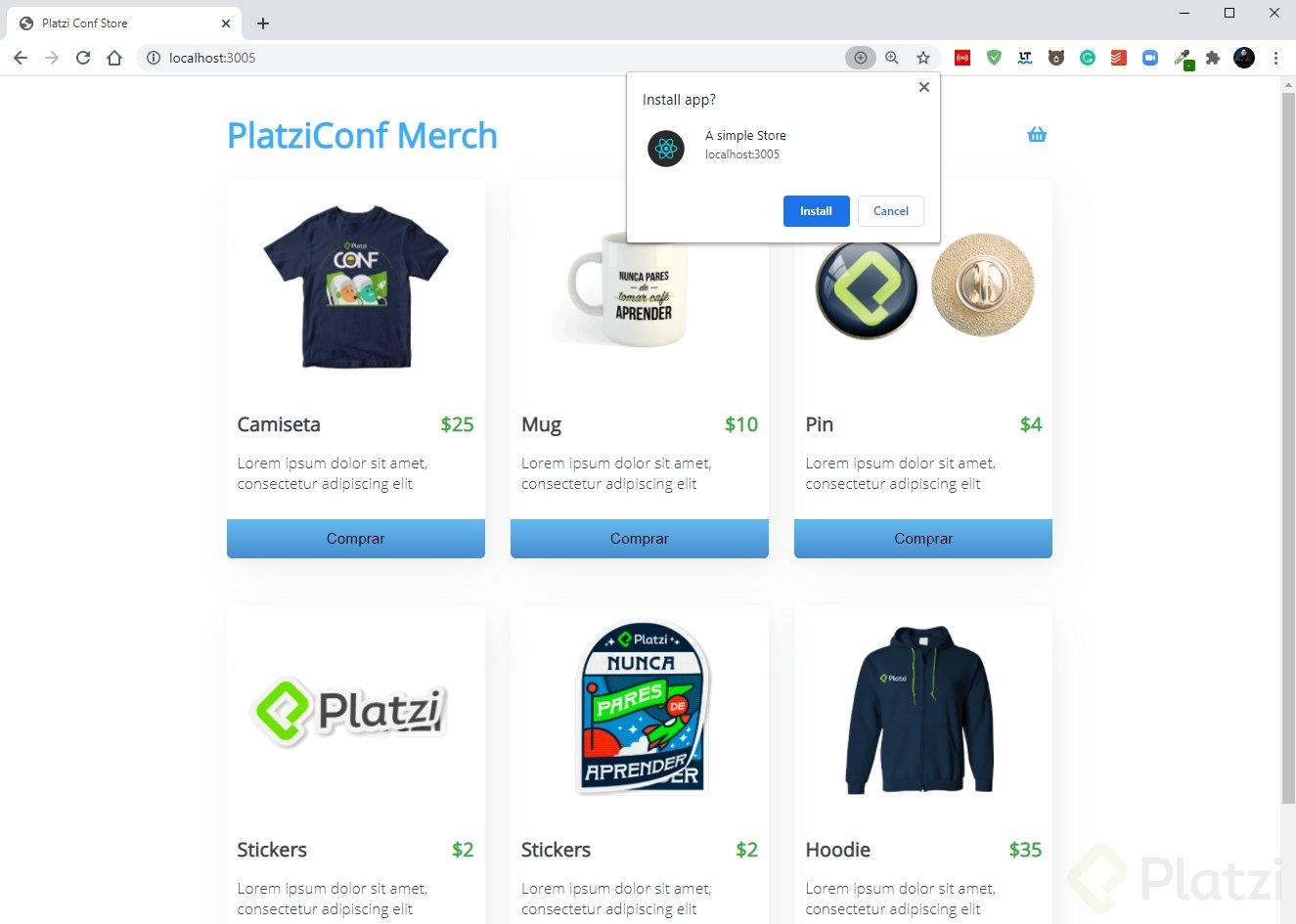
<link rel="manifest" href="/manifest.json">

​  
Una vez agregado el script y la referencia al manifest dentro del archivo index.html procedemos a compilar el proyecto:  
​

npm run build

​  
Cuando termina el proceso podemos revisar la carpeta /dist que ha generado Webpack y revisar que tenemos dentro el archivo manifest.json así como el archivo service-worker.js y dentro de assets el ícono que vamos a utilizar.  
​  
Ahora podemos correr el proyecto y comprobar que tenemos el registro del service worker y la información de nuestra aplicación:  
​

npm run start

​  
Service worker registrado correctamente:  
​  
  
​  
Registro del manifest dentro de la aplicación:  
​  
  
​  
Ahora puedes instalar Platzi Conf Store en tu computadora y aprovechar las ventajas de utilizar una PWA:  
​  
  
​  
Si quieres aprender más acerca de convertir tus aplicaciones de React en rápidas aplicaciones te recomiendo el curso [Curso de Progressive Web Apps con React.js](https://platzi.com/cursos/pwa-react-js/).