

## Guía de cómo hacer deploy de tu app, en github pages con React JS

Cuando ya tienes tu app terminada o un MVP (producto mínimo viable de tu app) y quieres tener su deploy para enseñarla.

1. Debes sacar un branch copia de máster, o una copia del branch en donde tu proyecto ya esté listo, que se vea bien cuando lo ves desde tu localhost.

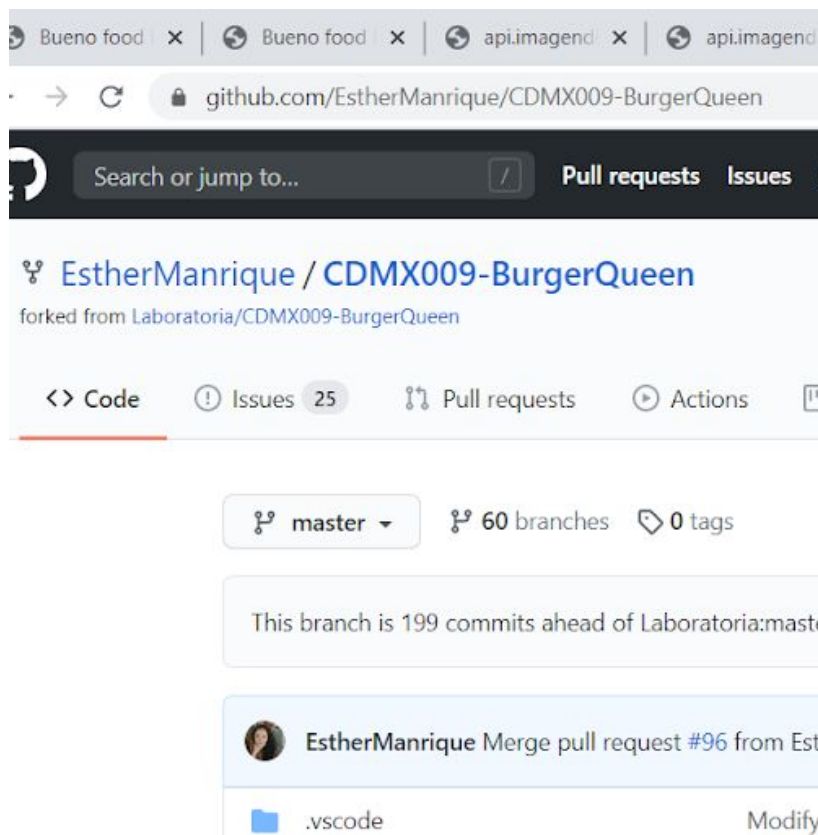
Un ejemplo del nombre que llevará tu branch podría ser:

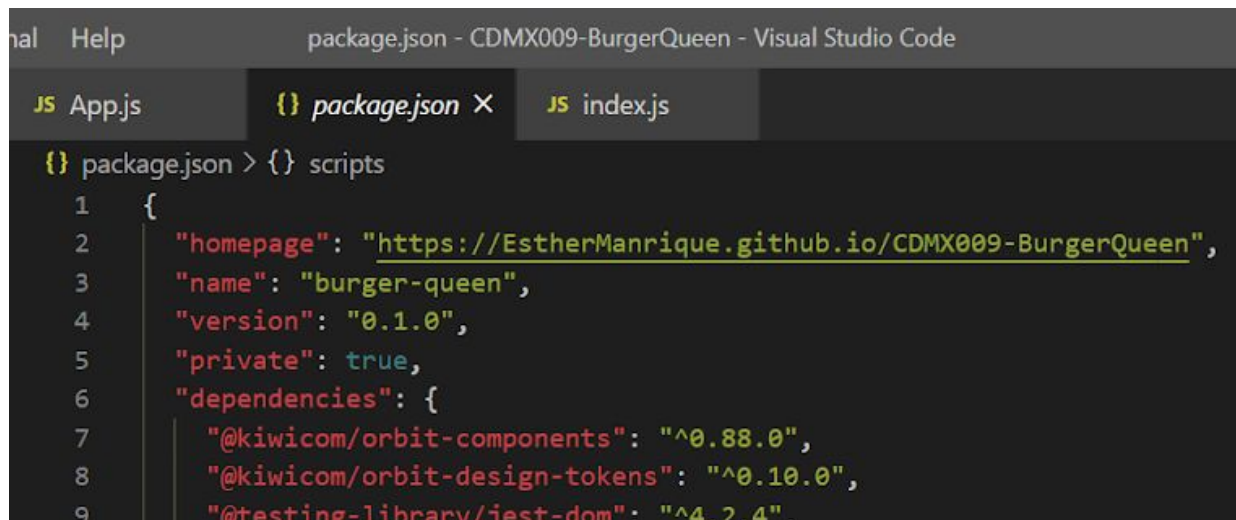
gh-pages-release-v-1.0 o 1.1. O el de tu preferencia es bueno que lleves un orden en cuanto a versiones, porque a futuro cuando hagas mejoras a tu página la versión ya sería 1.1 o 1.2, 1.3 y así sucesivamente porque cada release tiene sus cambios, es decir, el nuevo código que le ingresas y que se ve diferente en tu navegador. Ejemplo: si en la versión 1.1 haces deploy de un header, y luego codeas y ya luego estas codeando un slider en tu deploy 2.2 sería tal cual ese nuevo contenido.

2. estando en el nuevo branch que creaste para tu gh-pages vas a correr este comando: **npm install --save-dev gh-pages.**
3. Luego de eso en tu package.Json de tu proyecto debes agregar una propiedad que se llama **homepage**, en la línea 1 con la dirección de tu repo de github donde se aloja tu proyecto:

"homepage" : "<https://yourusername.github.io/repository-name>"

Un ejemplo más claro, te mostraré una captura de como lo hicimos:





```
1 {
2   "homepage": "https://EstherManrique.github.io/CDMX009-BurgerQueen",
3   "name": "burger-queen",
4   "version": "0.1.0",
5   "private": true,
6   "dependencies": {
7     "@kiwicom/orbit-components": "^0.88.0",
8     "@kiwicom/orbit-design-tokens": "^0.10.0",
9     "@testing-library/jest-dom": "^4.2.4",
```

4. En el mismo package Json te vas a la línea que dice scripts y agregas los siguientes:

```
"predeploy": "npm run build",
"deploy": "gh-pages -d build",
```

Una manera más gráfica, les mostraremos como lo hicimos en el nuestro:

```
1 {
2   "homepage": "https://EstherManrique.github.io/CDMX009-BurgerQueen",
3   "name": "burger-queen",
4   "version": "0.1.0",
5   "private": true,
6   "dependencies": {
7     "@testing-library/jest-dom": "^4.2.4",
8     "@testing-library/react": "^9.5.0",
9     "@testing-library/user-event": "^7.2.1",
10    "bootstrap": "^4.5.0",
11    "bootstrap-input-spinner": "^1.13.16",
12    "react": "^16.13.1",
13    "react-dom": "^16.13.1",
14    "react-router-dom": "^5.2.0",
15    "react-scripts": "3.4.1",
16    "styled-components": "^5.1.1"
17  },
18  "scripts": {
19    "predeploy": "npm run build",
20    "deploy": "gh-pages -d build",
21    "start": "react-scripts start",
```

Ese ejemplo es solo un pedacito de código para que sea una muestra más real del cómo ingresar los scripts. En la línea 2 podemos visualizar la ruta del homepage que fue uno de los pasos que hicimos, y en la línea 19 y 20 podemos ver los scripts que añadimos.

Ya puedes guardar los cambios del package y nuestro siguiente paso es en app.js norma

5. Ya puedes guardar los cambios del package y nuestro siguiente paso es en app.js normalmente el React Router viene así:

```
import React from "react";
import {
  BrowserRouter as Router,
  Switch,
  Route,
  Link
} from "react-router-dom";
```

Pero github pages no es compatible con el route **BrowserRouter**, por esto si queremos que el deploy de nuestra página funcione tenemos que cambiar ese BrowserRouter por **HashRouter**. Quedaría así:

```
1 import React, { useState } from 'react';
2 import{
3   Switch,
4   Route,
5   HashRouter}
6 from "react-router-dom";
```

Es importante que tengas en cuenta que luego de colocar el HashRouter en el import como está en el ejemplo de arriba, en líneas debajo de tu código en donde se encuentra tu BrowserRouter también lo reemplaces por HashRouter así como está en este ejemplo:

```
return (
  <div className="App">
    <HashRouter>
      <Switch>
        <Route exact path="/">
          <BackgLogin />
        </Route>
      </Switch>
    </HashRouter>
  </div>
);
}
```

6. Luego de hacer con detalle estos cambios, y que te hayas detenido a revisar que todo esté ingresado y reemplazado correctamente guardas los cambios, revisas que tu app siga corriendo, la cual debería hacerlo sin problema. y haces tu commit y push de los cambios que hiciste.
7. El paso siguiente es que en tu terminal allí mismo dentro de tu proyecto vas a correr los siguientes comandos en este mismo orden:  
el primero: `npm run build`  
el segundo: `npm run deploy`
8. Al correr estos comandos se van a generar unos cambios, esto es normal ya que se generaron chunks que es como tu código comprimido esto es parte del proceso pero sobre todo de correr estos comandos, también tu link del deploy de tu página está a punto de salir del horno, tan solo haces commit y push, te vas a tu package.json y en la línea 2 en el homepage donde ingresaste la ruta de tu repositorio copia solo lo que está dentro de las comillas y eso te lo llevas al buscador de tu navegador, lo pegas, das enter y ya debería salir tu deploy.

Recuerda: Estos comandos solo debes correrlos en tu branch de gh-pages-v en el que estás trabajando, lo último que debes hacer es hacer merge de este branch con tu master para que estén iguales por los chunks que se generaron con estos comandos.

Aquí está el ejemplo del deploy de nuestra app y la hicimos con estos pasos que te acabamos de compartir, debes tener en cuenta que la app por el momento está diseñada para IPAD-PRO, por lo que cuando entres al link da click al boton derecho, seleccionas inspeccionar y en el select que dice RESPONSIVE selecciona IPAD-PRO.  
<https://esthermanrique.github.io/CDMX009-BurgerQueen/#/>

Si te fijas al final de tu link o del que te pasamos, sale un `/#/` no te preocupes es parte del HashRouter (Hash es como hashtag por eso sale el `#` )

Cualquier duda nos puedes preguntar, andale, deja la pena de lado o escondela en tu bolsillo.

En discord o en el correo

Esther Manrique: [esmango0610@gmail.com](mailto:esmango0610@gmail.com)

Dianyela Maldonado: [dianyela.y.maldonado@gmail.com](mailto:dianyela.y.maldonado@gmail.com)