

## CONFIGURACIÓN WEBPACK Y BABEL

### 1.) Creamos el package.json

```
npm init --Y
```

### 2.) Creamos la siguiente estructura de carpetas

```
src
```

```
src/app
```

```
src/app/index.js
```

```
src/index.html
```

### 3.) Instalamos las siguientes dependencias

```
npm install webpack webpack-cli @babel/core @babel/cli @babel/preset-env @babel/polyfill  
babel-loader html-webpack-plugin webpack-dev-server css-loader style-loader mini-css-extract-  
plugin
```

### 4.) En la raíz del Proyecto creamos un archivo llamado .babelrc con el siguiente código

```
{  
  "presets": ["@babel/preset-env"]  
}
```

### 5.) En la raíz creamos un archivo llamado webpack.config.js con el siguiente código

```
const path = require('path');  
  
const HTMLWebpackPlugin = require('html-webpack-plugin')  
const MiniCssExtractPlugin = require('mini-css-extract-plugin');
```

```
module.exports = {  
  mode: 'none',  
  entry: {  
  
    app: ["@babel/polyfill", './src/app/index.js']  
  },  
  output: {
```

```
    path: path.resolve(__dirname, 'build'),
    filename: 'js/app.bundle.js'
  },
  devServer: {
    port: 5050
  },
  module: {
    rules: [
      {
        test: /\.js$/i,
        loader: 'babel-loader'
      },
      {
        test: /\.css$/i,
        use: [MiniCssExtractPlugin.loader, 'css-loader'],
      }
    ]
  },
  plugins: [
    new HTMLWebpackPlugin({
      template: './src/index.html',
      minify: {
        collapseWhitespace: true,
        removeComments: true,
        removeRedundantAttributes: true,
        removeScriptTypeAttributes: true,
        removeStyleLinkTypeAttributes: true,
        useShortDoctype: true
      }
    })
  ]
}
```

```

    }},
    new MiniCssExtractPlugin({
      filename: 'css/app.bundle.css'
    }),
  ]
};

```

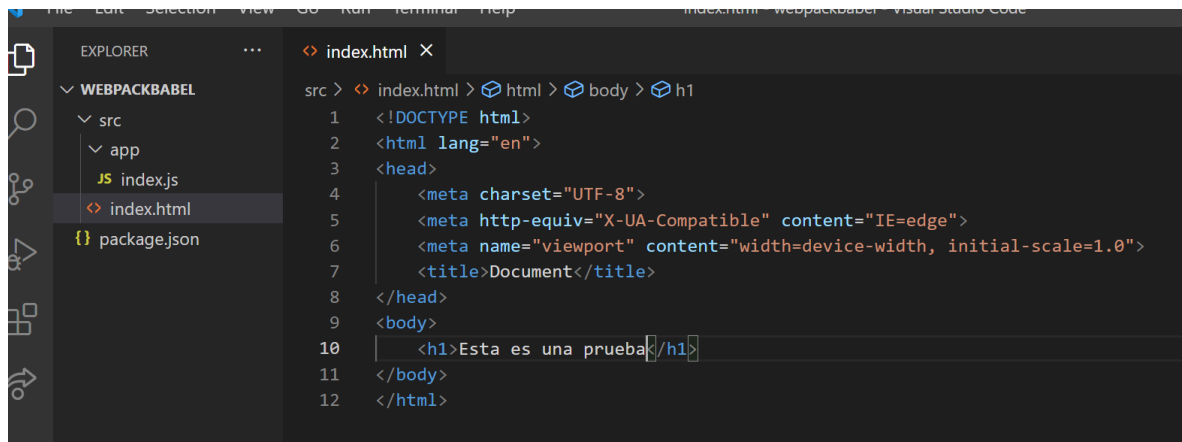
6.) En el package.json realizamos la siguiente adición en los scrips

```

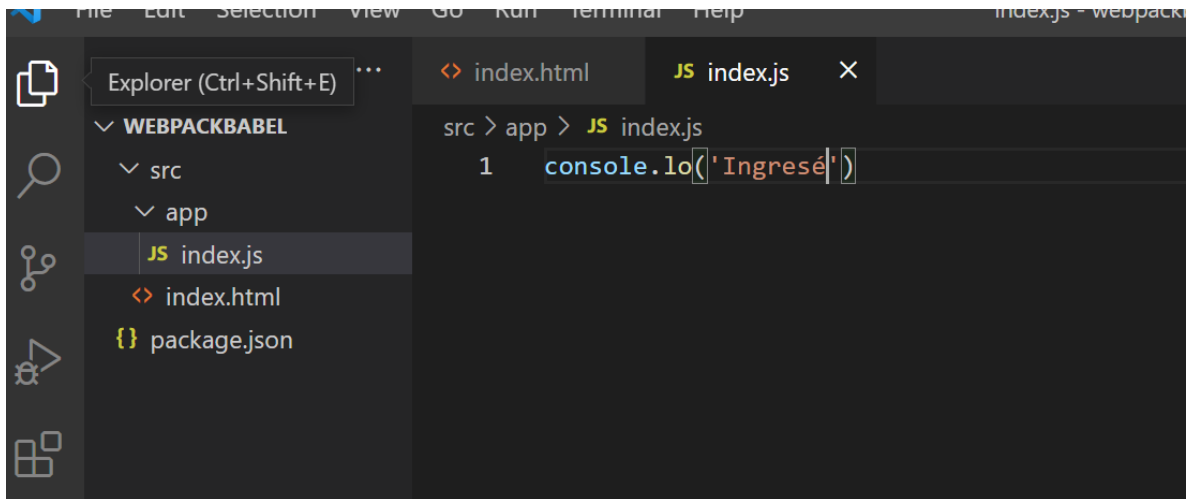
"start": "webpack --mode development",
"build": "webpack --mode production",
"dev": "webpack-dev-server --open"

```

7.) En el archivo index.html insertamos una estructura base de html



8.) En el archivo index.js insertamos un console.log de prueba



**9.) Probaremos. Corremos el proyecto en modo desarrollo, luego en modo producción y luego levantamos el servidor con webpack server**

npm run start

npm run dev

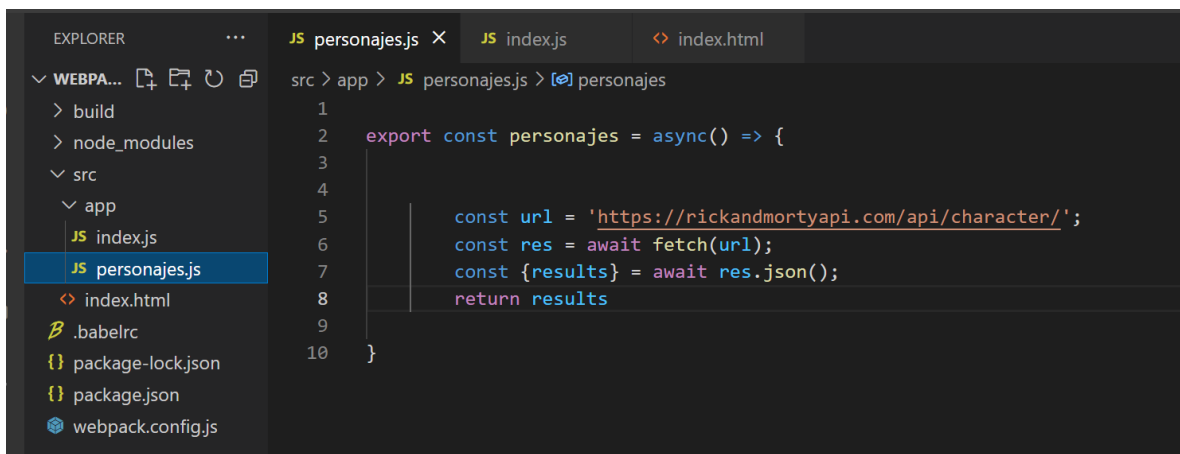
verificamos los resultados en el navegador

Eliminamos la carpeta que se generó llamada build

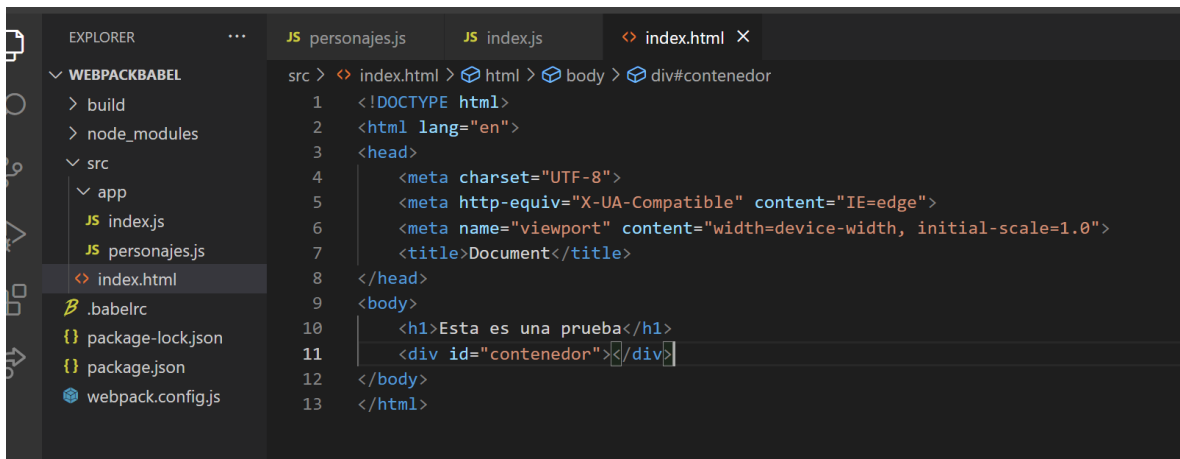
npm run build y corremos el archivo index.html con live server

**10.) Ahora vamos a consumirnos un API <https://rickandmortyapi.com/api/character/>**

Dentro de la carpeta app creamos un archivo llamado personajes.js



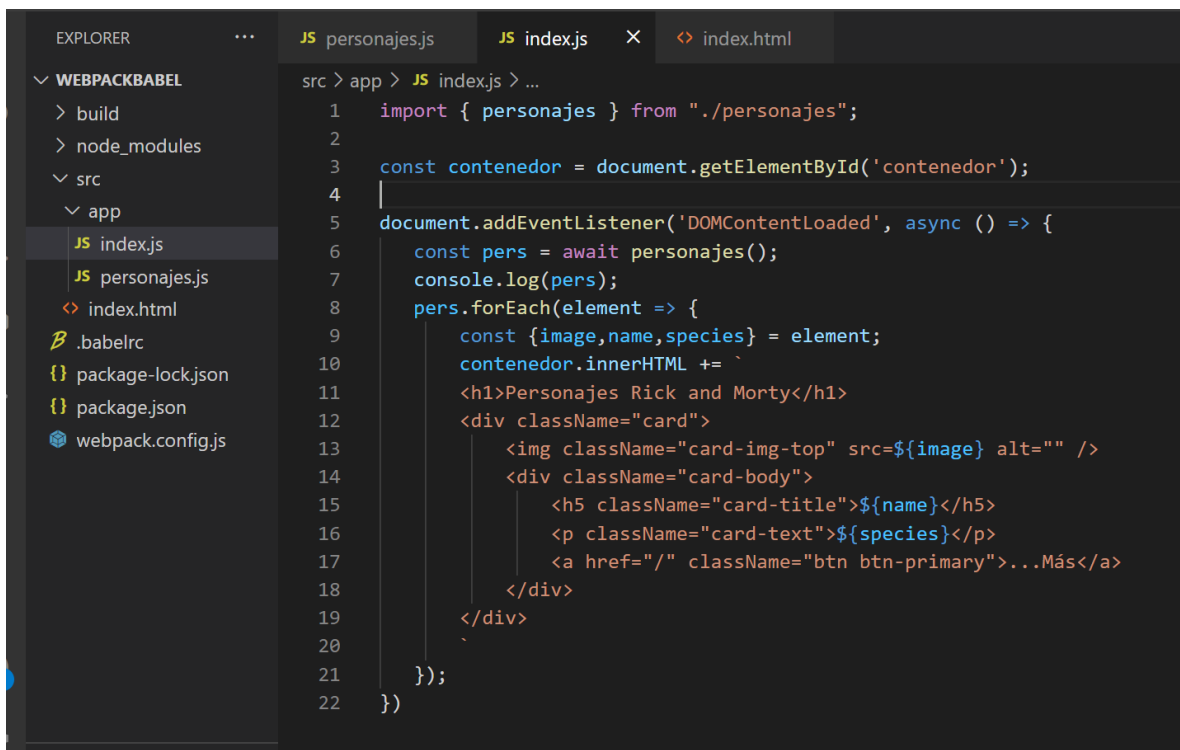
Dentro del archivo index.html creamos una división y le asignamos un id



The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer shows a project structure with folders 'build', 'node\_modules', and 'src'. Inside 'src', there is an 'app' folder containing 'index.js', 'personajes.js', and 'index.html'. The 'index.html' file is selected and its content is displayed in the editor. The HTML code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10   <h1>Esta es una prueba</h1>
11   <div id="contenedor"></div>
12 </body>
13 </html>
```

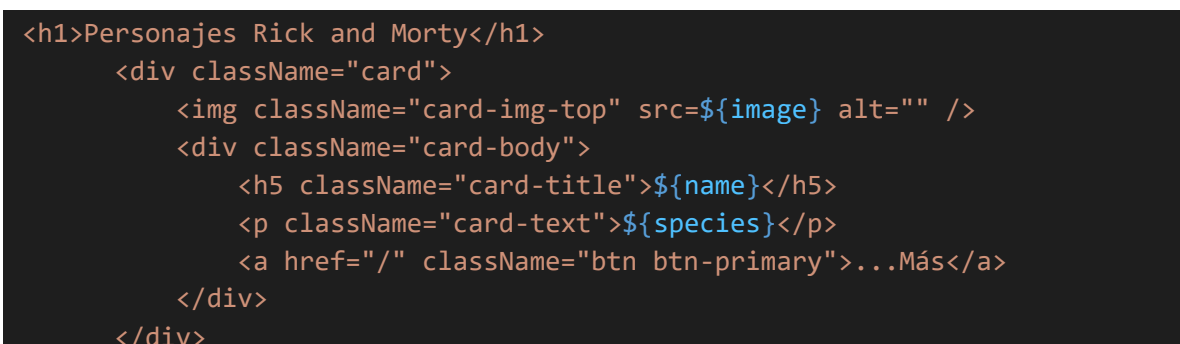
En el archivo index llamamos la función que creamos en personajes.js



The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer shows the same project structure as before. The 'index.js' file is selected and its content is displayed in the editor. The JavaScript code is as follows:

```
1 import { personajes } from './personajes';
2
3 const contenedor = document.getElementById('contenedor');
4
5 document.addEventListener('DOMContentLoaded', async () => {
6   const pers = await personajes();
7   console.log(pers);
8   pers.forEach(element => {
9     const {image,name,species} = element;
10    contenedor.innerHTML += `
11    <h1>Personajes Rick and Morty</h1>
12    <div className="card">
13      <img className="card-img-top" src=${image} alt="" />
14      <div className="card-body">
15        <h5 className="card-title">${name}</h5>
16        <p className="card-text">${species}</p>
17        <a href="/" className="btn btn-primary">...Más</a>
18      </div>
19    </div>
20    `;
21  });
22 })
```

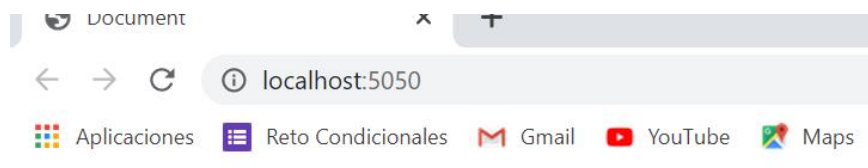
Editable



The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer shows the same project structure as before. The 'index.html' file is selected and its content is displayed in the editor. The HTML code is as follows:

```
<h1>Personajes Rick and Morty</h1>
<div className="card">
  <img className="card-img-top" src=${image} alt="" />
  <div className="card-body">
    <h5 className="card-title">${name}</h5>
    <p className="card-text">${species}</p>
    <a href="/" className="btn btn-primary">...Más</a>
  </div>
</div>
```

## Resultados



**Esta es una prueba**

## Personajes Rick and Morty



**Rick Sanchez**

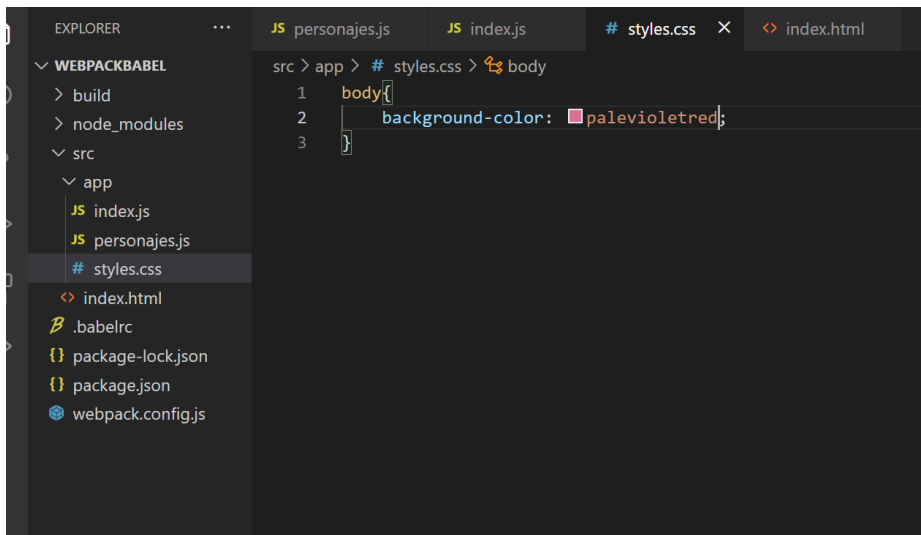
Human

[...Más](#)

## Personajes Rick and Morty

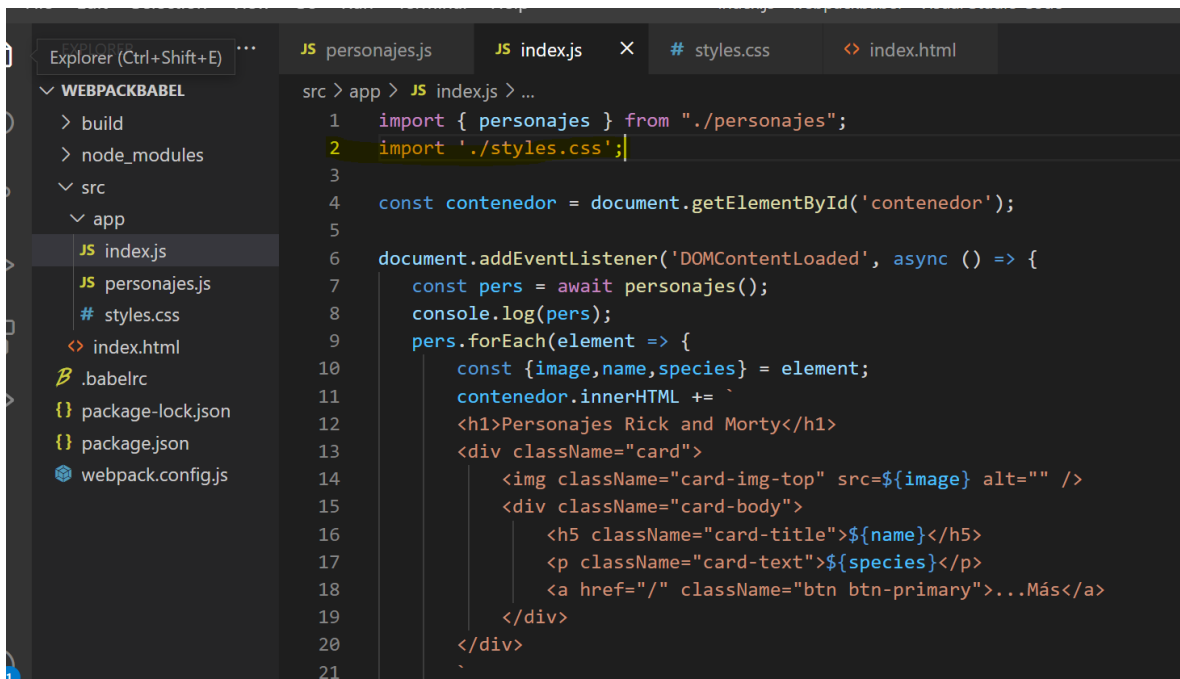


Creamos un archivo llamado styles.css dentro de app



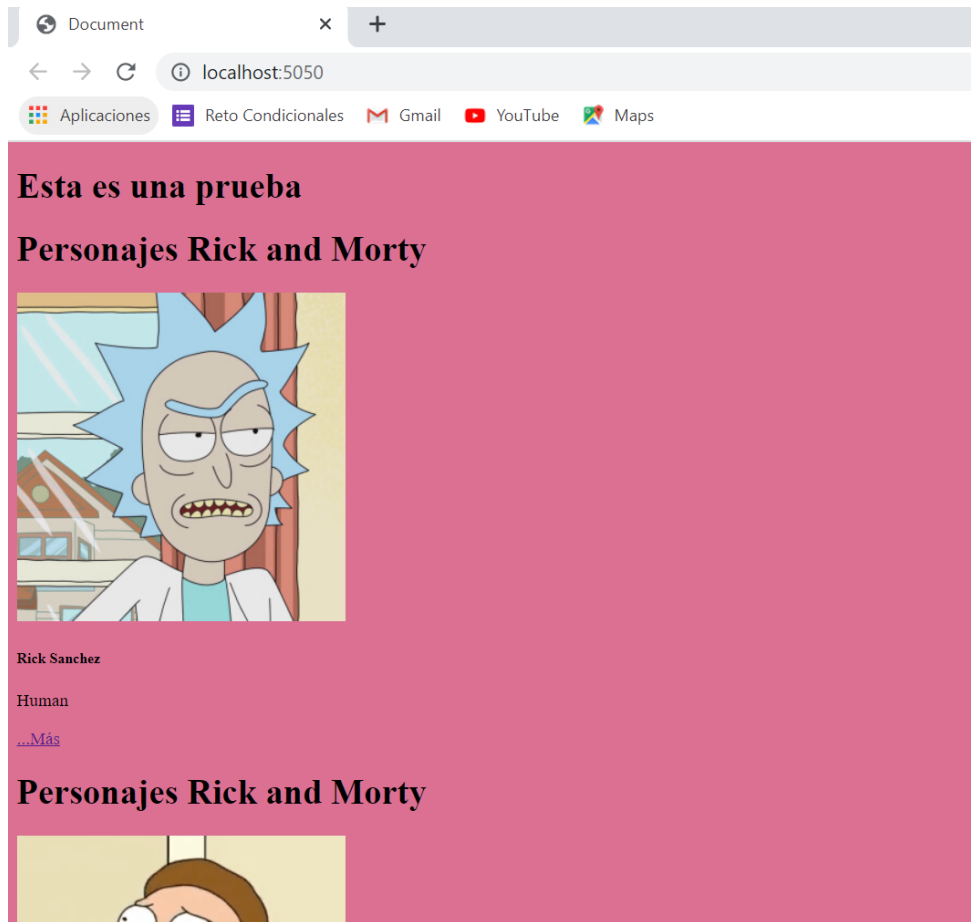
```
src > app > # styles.css > body
1 body{
2   background-color: palevioletred;
3 }
```

Este archivo lo importamos dentro del archivo index.js

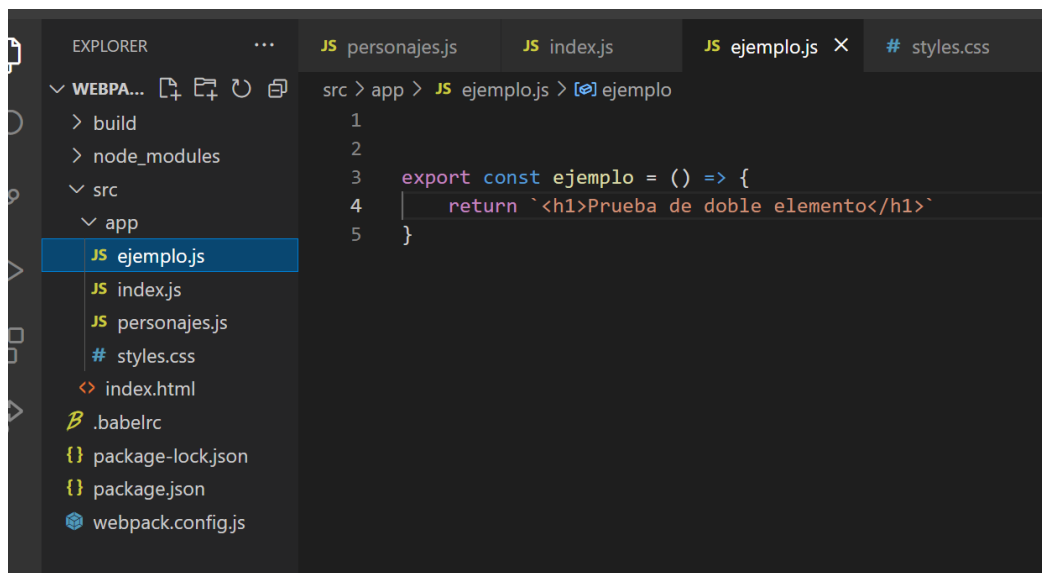


```
src > app > JS index.js > ...
1 import { personajes } from "../personajes";
2 import './styles.css';
3
4 const contenedor = document.getElementById('contenedor');
5
6 document.addEventListener('DOMContentLoaded', async () => {
7   const pers = await personajes();
8   console.log(pers);
9   pers.forEach(element => {
10     const {image,name,species} = element;
11     contenedor.innerHTML += `
12       <h1>Personajes Rick and Morty</h1>
13       <div className="card">
14         <img className="card-img-top" src=${image} alt="" />
15         <div className="card-body">
16           <h5 className="card-title">${name}</h5>
17           <p className="card-text">${species}</p>
18           <a href="/" className="btn btn-primary">...Más</a>
19         </div>
20       </div>
21     `
```

Resultados



Ahora adicionaremos dentro de app un archivo llamado ejemplo.js



Y lo pintaremos en pantalla también, relacionándolo en index.js



```
src > app > JS index.js > document.addEventListener('DOMContentLoaded') callback > pers.forEach() callback
1 import { personajes } from "../personajes";
2 import { ejemplo } from "../ejemplo";
3 import './styles.css';
4
5 const contenedor = document.getElementById('contenedor');
6
7 document.addEventListener('DOMContentLoaded', async () => {
8   contenedor.innerHTML += ejemplo();
9   const pers = await personajes();
10  console.log(pers);
11  pers.forEach(element => {
12    const {image,name,species} = element;
13    contenedor.innerHTML += `
14    <h1>Personajes Rick and Morty</h1>
15    <div className="card">
16      <img className="card-img-top" src=${image} alt="" />
17      <div className="card-body">
18        <h5 className="card-title">${name}</h5>
19        <p className="card-text">${species}</p>
20        <a href="/" className="btn btn-primary">...Más</a>
21      </div>
```

Resultado

**Esta es una prueba**

**Prueba de doble elemento**

**Personajes Rick and Morty**



Rick Sanchez

Human

[...Más](#)

**Personajes Rick and Morty**



Antes de subir al repositorio en la raíz del proyecto crea un archivo llamado .gitignore y dentro relacionemos los node\_modules

