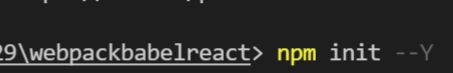
**CONFIGURACIÓN WEBPACK Y BABEL CON REACT**

**Repositorio ejercicio** [**https://github.com/jennymontoya1001/webpackbabelreact.git**](https://github.com/jennymontoya1001/webpackbabelreact.git)

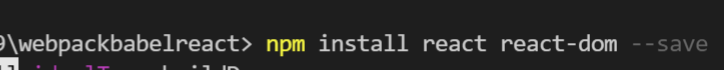
**1.) En VSC abrimos la terminal e instalamos el package.json**

npm init --Y



**2.) Vamos a instalar unas dependencias para poder trabajar con react**

npm install react react-dom --save



**3.) Crearemos nuestra estructura de carpetas**

public

public/index.html

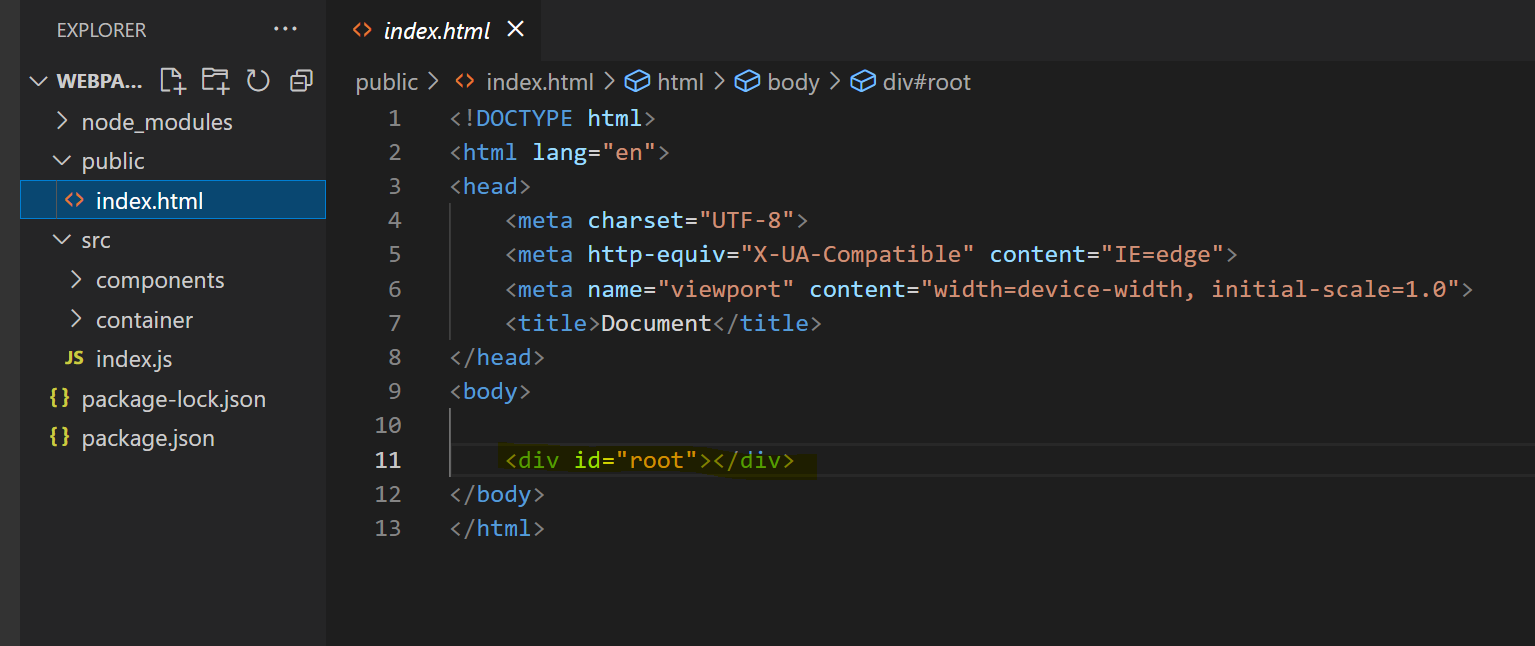
src

src/components

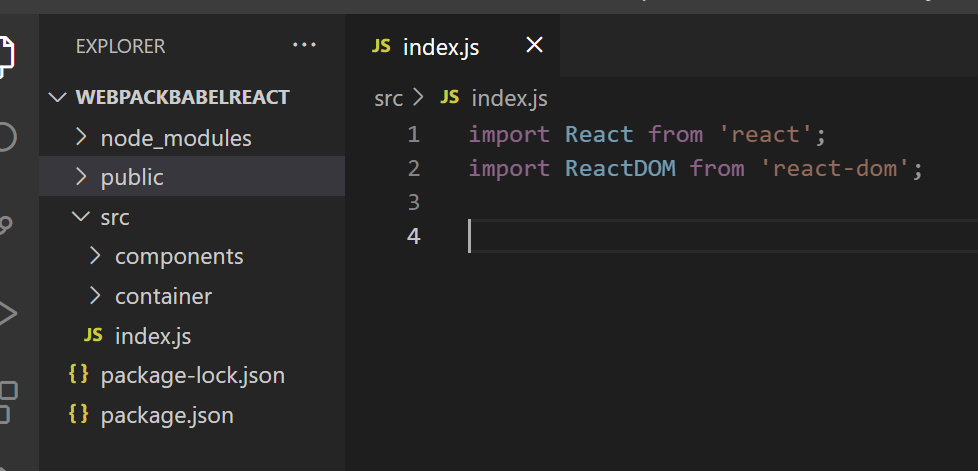
src/containers

src/index.js

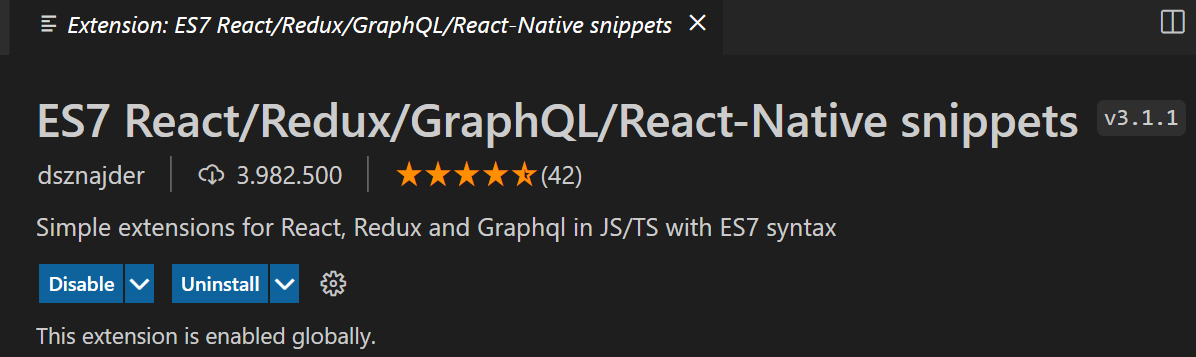
**4.) Dentro del archive index.html que se encuentra en la carpeta public, insertamos una estructura base de html y dentro del body insertamos un div con un id llamado root**



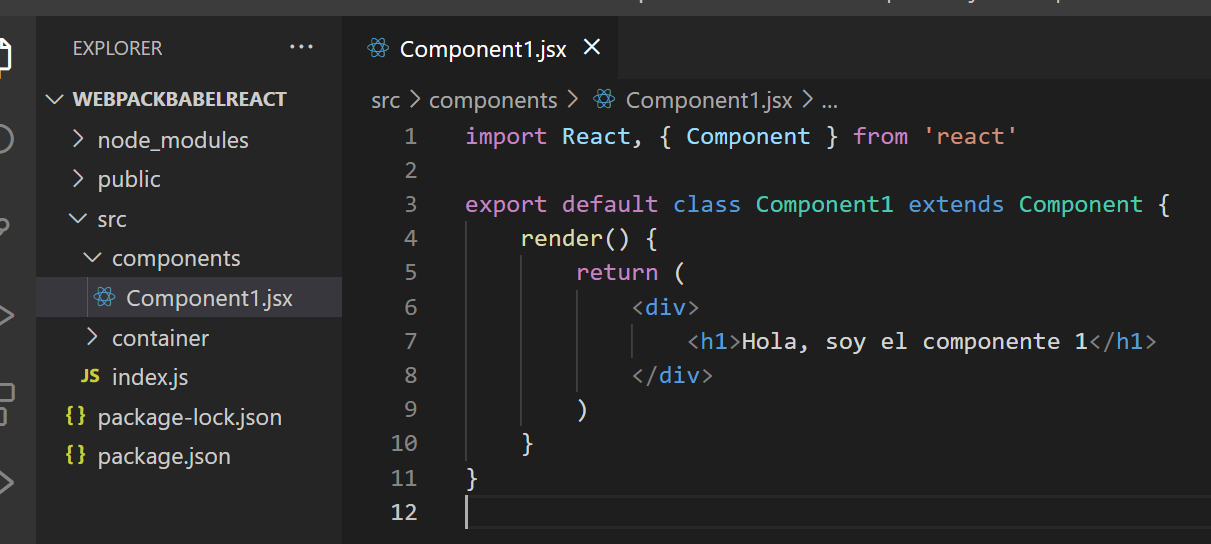
**5.) Nos vamos para la carpeta src y abrimos el archivo index.js. Importamos las librerías de react y reac-dom**



**6.) Adicionamos la siguiente extensión**



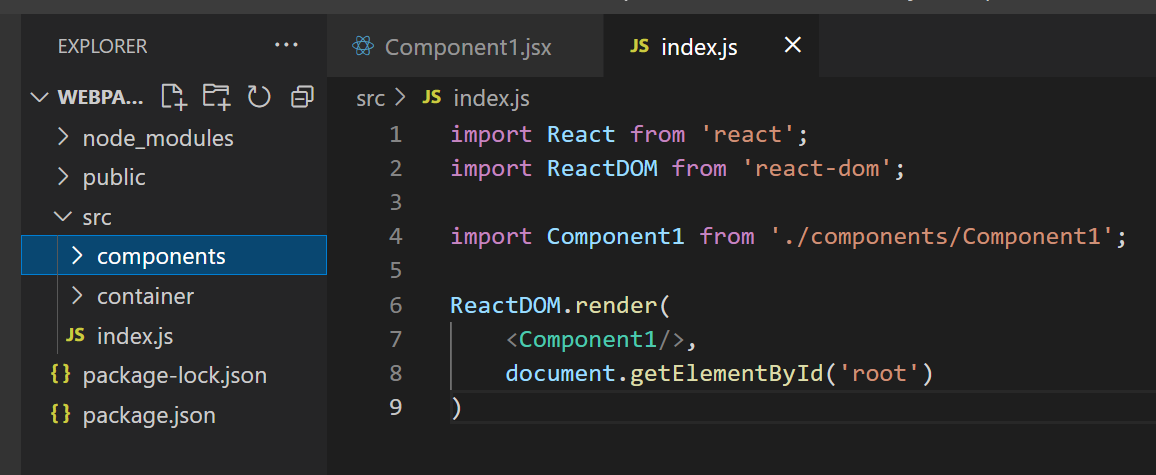
**7.) Dentro de la carpeta component creamos uno llamado Component1.jsx**



**8.) Vamos a implementar el componente para poderlo ver. La implementación de un componente es convertirlo en un elemento o etiqueta**

**En el archivo index.js importamos a Component1.js**

**Como elemento de entrada usamos el div con id root definido en el archivo index.html de la carpeta public**



**9.) Ingresamos al siguiente link. En él encontraremos lo necesario para configurar webpack y babel** <https://gist.githubusercontent.com/jennymontoya1001/4cda0e4d2014b0b7d512a4cd94e9c99b/raw/905e4e9c8e4eaa4c7f09b24dbe761e2540285f2b/ReactWebpackBabel>

Empezaremos a configurar babel

Instalamos las siguientes dependencias

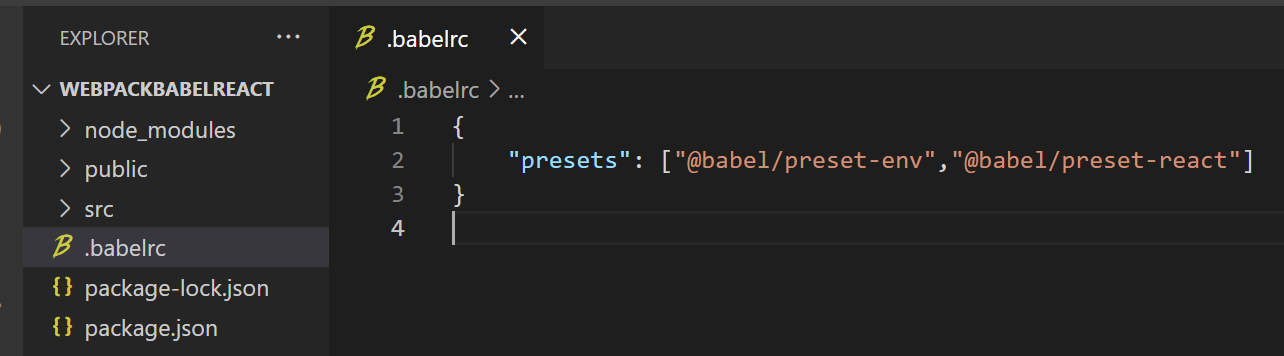
npm install @babel/core @babel/preset-react @babel/preset-env babel-eslint babel-loader --save-dev

**10.) En la raíz del Proyecto creamos un archive llamado .babelrc**

{

    "presets": ["@babel/preset-env","@babel/preset-react"]

}



**11.) Ahora instalaremos webpack**

npm install webpack webpack-cli html-webpack-plugin html-loader mini-css-extract-plugin css-loader --save-dev

**12.) Vamos a instalar otra dependencia de desarrollo que nos sirve para cargar nuestra aplicación en local webpack dev server**

npm i -D webpack-dev-server

**13.) Dentro de la carpeta raíz, a la misma altura de src creamos un archivo llamado webpack.config.js**

//configurar nuestro path

const path = require('path');

//plugin que instalamos

const HtmlWebPackPlugin = require('html-webpack-plugin');

const MiniCssExtractPlugin = require('mini-css-extract-plugin');

module.exports = {

  //entrada , cual va a ser mi elemento principal, punto de entrada

  entry: './src/index.js',

  //a donde quiero empujar mi proyecto ouput

  output: {

    //saber donde me encuentro, directorio donde esta el proyecto

    //y ahi crea una carpeta dist donde se va a guardar nuestro

    //proyecto

    path: path.resolve(\_\_dirname, 'build'),

    //lamar a mi compilado de JS una asignacion de nombre, como se va

    //a construir

    filename: 'bundle.js',

  },

  //configuracion para saber con que archivos vamos a trabajar

  resolve: {

    //extenciones con las cuales estamos trabajando

    extensions: ['.js', '.jsx'],

  },

  //modulo donde vamos a tener las reglas de negocio del proyecto

  module: {

    //definimos las reglas por medio de arreglos

    rules: [

      {

        //creamos expresion regular para identificar estos

        //archivos

        test: /\.(js|jsx)$/,

        //vamos a excluir la carpeta de node modules

        exclude: /node\_modules/,

        //y vamos a utilizar el loder que utilizamos en babel

        use: {

          loader: 'babel-loader',

        },

      },

      //identificar los archivos html de nuestro proyecto, entenderlos

      //y prepararlos

      {

        test: /\.html$/,

        //uso del loader que previamente instalamos

        use: {

          loader: 'html-loader',

        },

      },

      {

        test: /\.css$/,

        use: [

          {

            loader: MiniCssExtractPlugin.loader,

          },

          'css-loader',

        ],

      },

    ],

  },

  //agregar plugin que instalamos para entender el archivo index y el que

  //vamos a generar en la carpeta dist para enviar a produccion

  plugins: [

    new HtmlWebPackPlugin({

      template: './public/index.html',

      //se va a empujar como

      filename: './index.html',

    }),

    new MiniCssExtractPlugin({

      filename: 'assets/[name].css',

    }),

  ],

  devServer: {

    static: path.join(\_\_dirname, 'build'),

    compress: true,

    //Habilitar para poder acceder a la información y mostrar cada una de las rutas

    historyApiFallback: true,

    port: 3005,

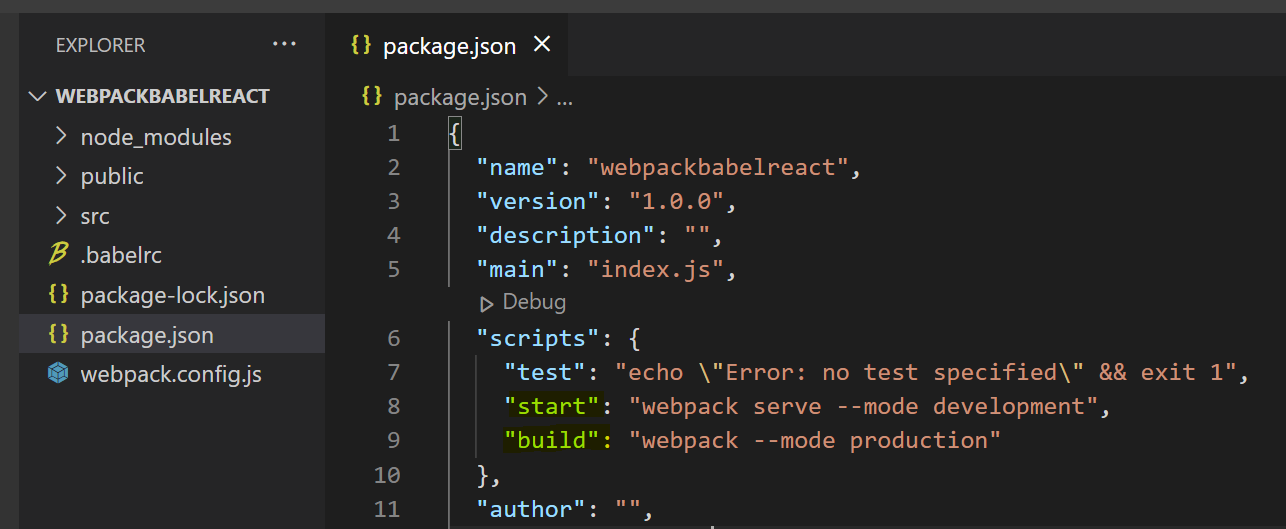
  },

};

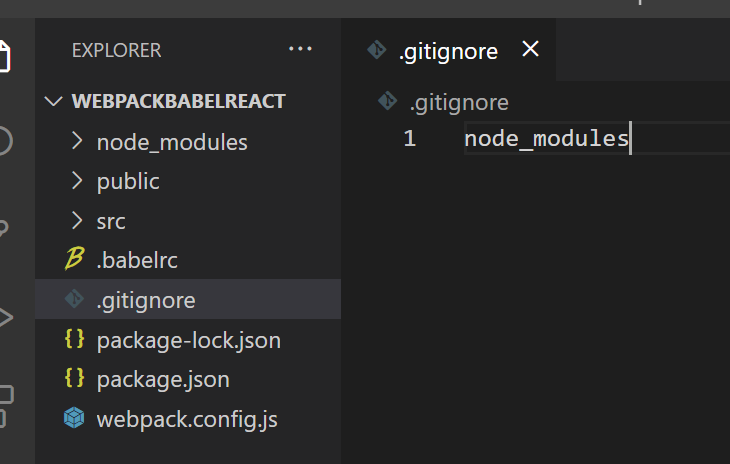
**14.) Nos vamos para nuestro paquete json y nos ubicamos en la zona de los scripts**

 "start": "webpack serve --mode development",

    "build": "webpack --mode production"



**Adicional en la raíz del proyecto un archivo llamado .gitignore**



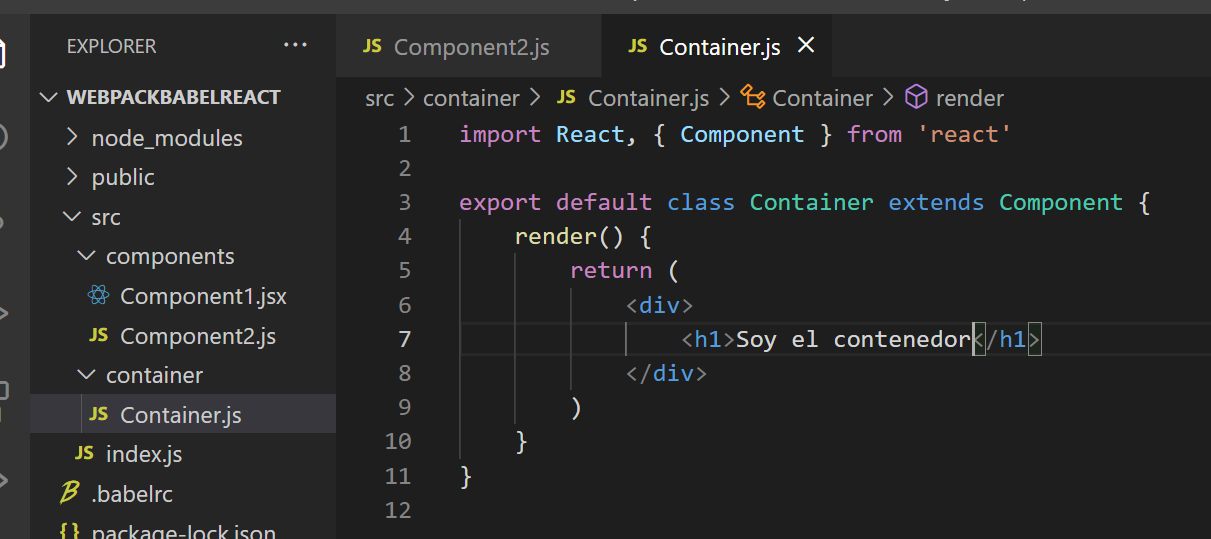
**15.) Correr la instrucción npm run start o nom start**



**16.) Dentro de la carpeta components crearemos Component2.js**

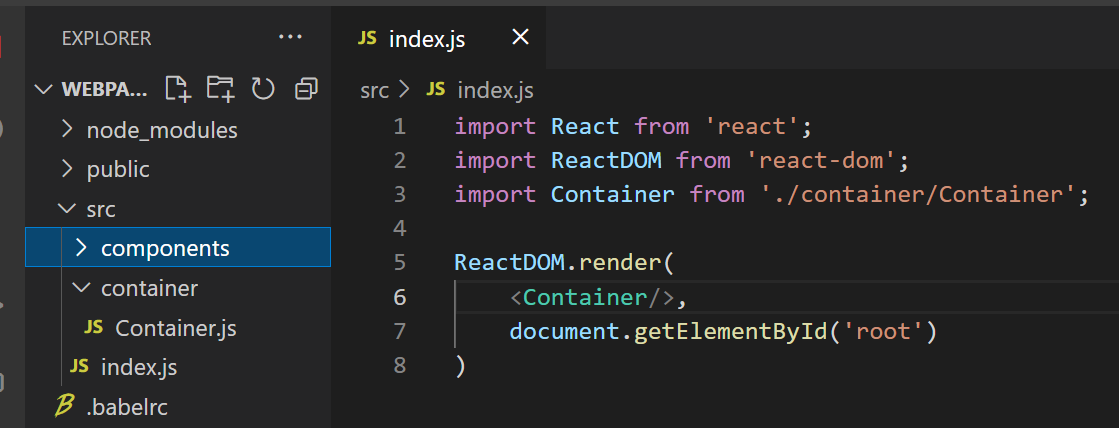


**17. Dentro de la carpeta containers crear uno llamado Container.js**

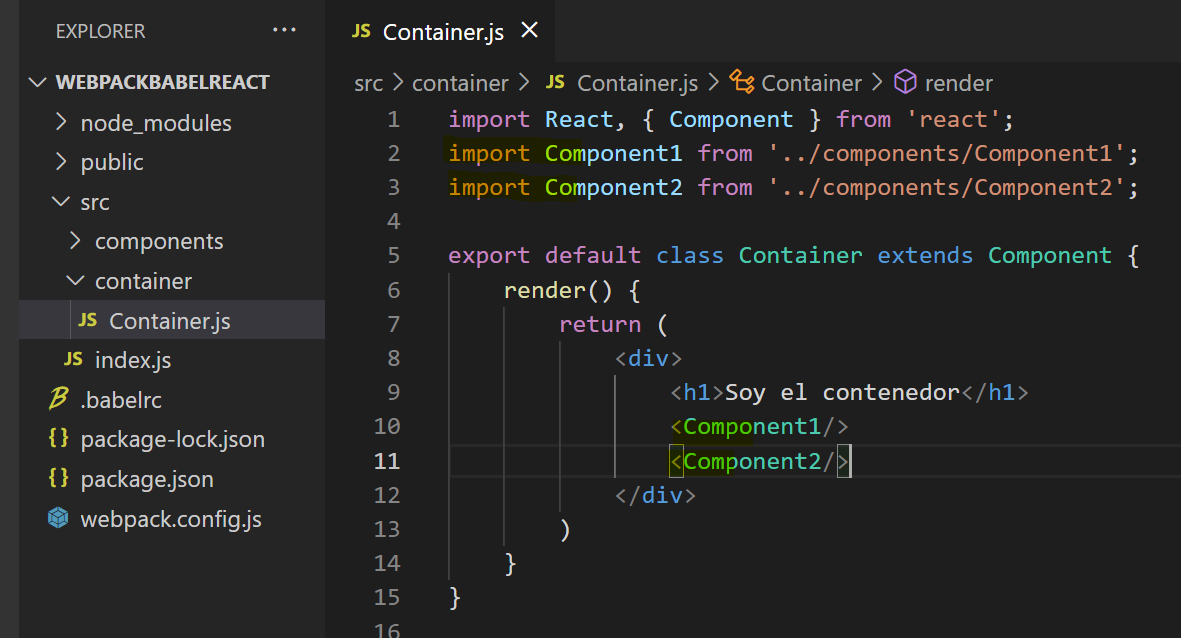


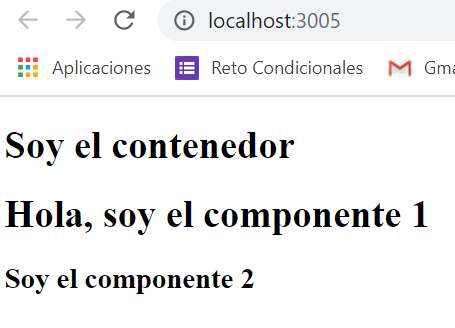
**18. El componente Container.js será el contenedor padre, Component1.js y Compoentn2.js serán los componentes hijos. Realizaremos los siguientes cambios**

**Dentro del archivo index.js importaremos Containers.js**



**19. Dentro de Container.js importamos Component1.js y Component2.js**

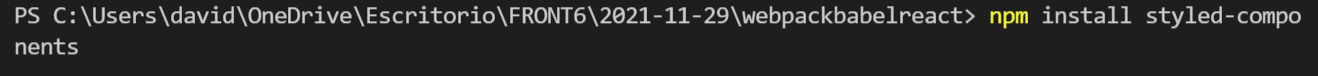




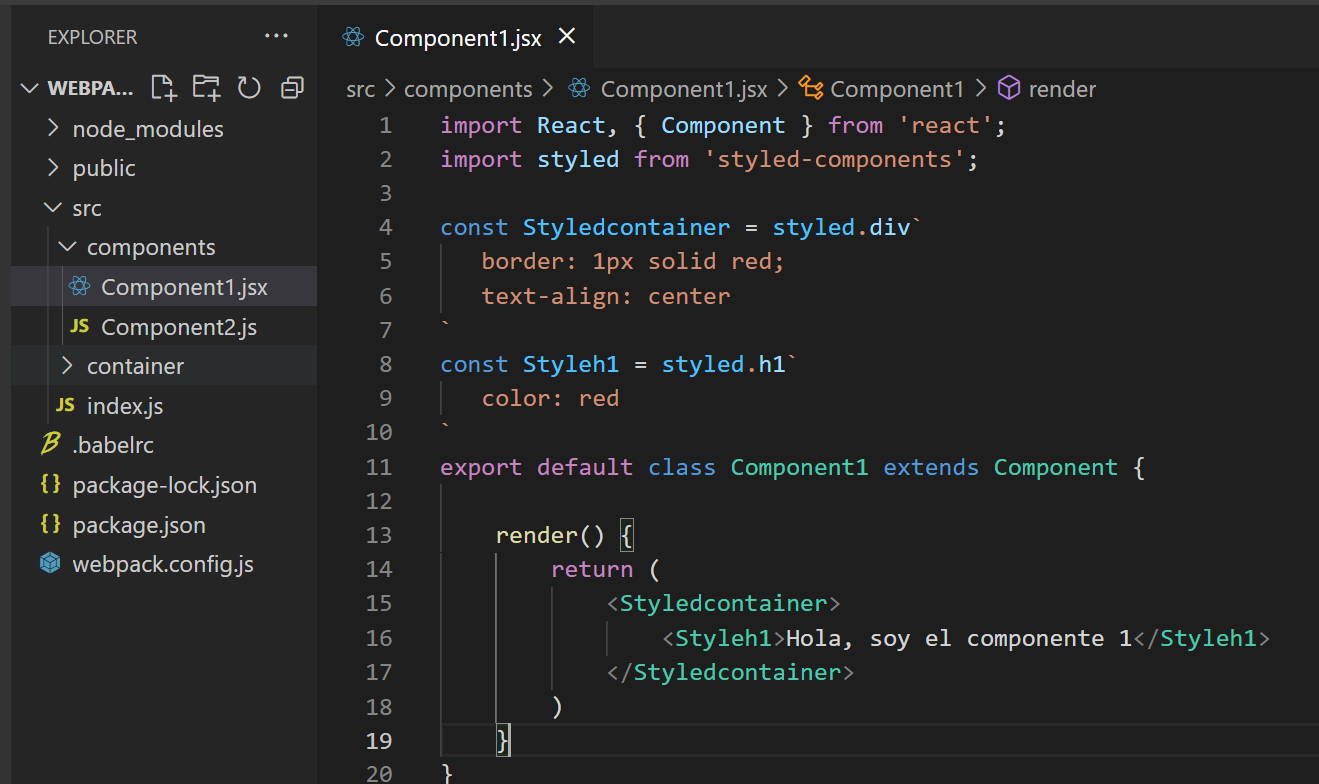
**STYLES COMPONENTS**

**Instalaremos una librería que nos permitirá trabajar con CSS puro**

npm install styled-components

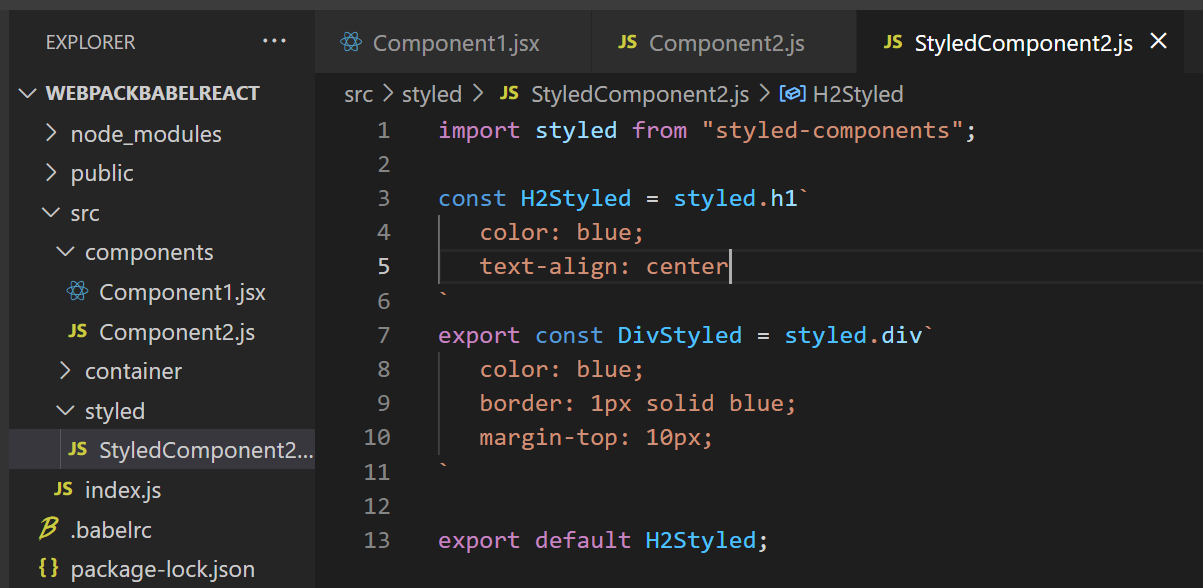


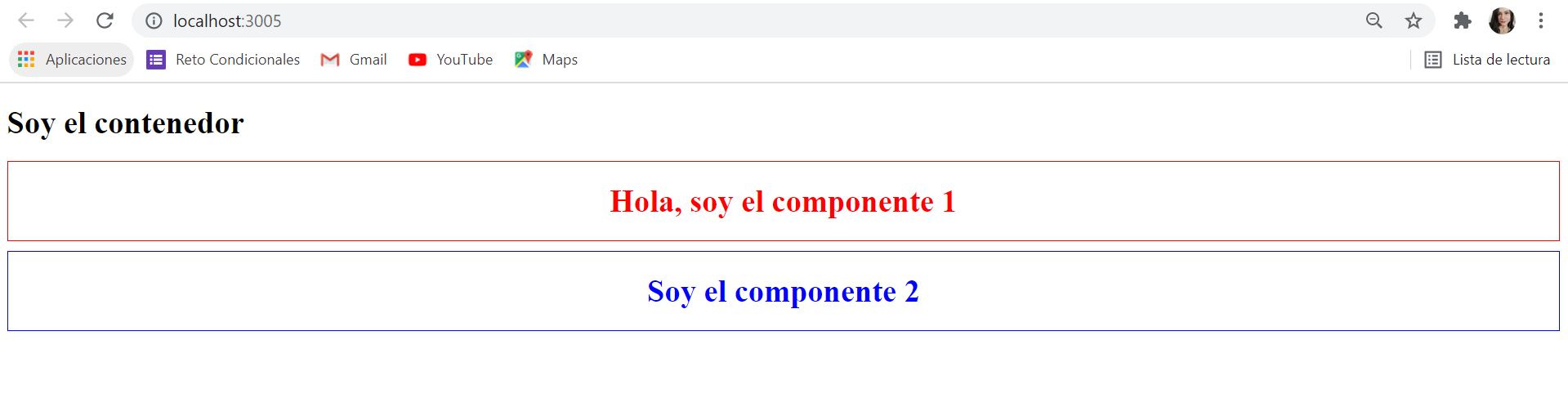
Component1.jsx



Component2.js

Creamos dentro de src una carpeta llamada styled, dentro de ella creamos un archivo llamado StyledComponent2.js





Container.js

