Creando proyecto

En esta sección vamos a comenzar a construir una aplicación tipo contador utilizando un hook importante en React, **useState**, para mantener el estado de la aplicación; además con ese hook se puede guardar y rescatar el último valor actualizado de un contador.

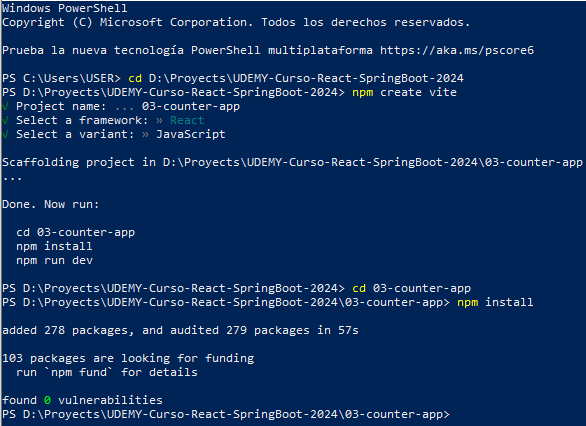
Introducción a los Hooks en React

* Los hooks se introdujeron en la versión 16 de React.
* Ofrecen diferentes características para manejar el estado, la lógica de un componente y los datos de un solo punto en la aplicación.
* React se utiliza para crear para aplicaciones del lado del cliente (frontend) de una sola página.
* No solo se encarga de la lógica del negocio, sino que también permite renderizar cambios de manera eficiente, actualizando solo los elementos afectados.
* Existe un hook para manejar el ciclo de vida de un componente, que se dispara en diferentes etapas como la inicialización y cambios de estado.
* Ejemplos de usos con **useState**: guardar información en un contador y un carrito de compras en una página web.

Comandos para crear un nuevo proyecto

Se tienen en cuenta los comandos que se utilizaron previamente para crear un nuevo proyecto desde la terminal PowerShell:

1. Utiliza el comando **cd** para navegar a la ubicación donde deseas crear el nuevo proyecto (también puedes hacer clic derecho para pegar una ruta copiada)
2. Ejecuta **npm create vite** para crear un nuevo proyecto con Vite.
3. Luego, ejecuta **npm install** para instalar las dependencias del proyecto luego de haber ejecutado el comando **cd** para navegar a la carpeta que contiene el proyecto.



**Configuración** inicial

Una vez creado el proyecto, sigue estos pasos (ya se utilizaron previamente en proyectos anteriores):

1. Abre Visual Studio Code.
2. Ve al menú **Archivo**, opción **Cerrar Carpeta** para cerrar cualquier proyecto anterior.
3. Arrastra la carpeta **03-counter-app** al editor de Visual Studio Code para abrir el proyecto.

Ejecución de la aplicación

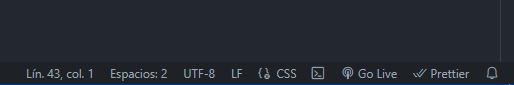
Sigue estos pasos para ejecutar la aplicación, en el caso de que no cargue aplicación web, intenta guardar los cambios en cada uno de los archivos y vuelve a recargar la página web.

1. Abre una nueva terminal haciendo clic en el menú **Terminal**, opción **Nueva Terminal**.
2. Escribe **npm run dev** en la terminal y presiona Enter para ejecutar la aplicación.

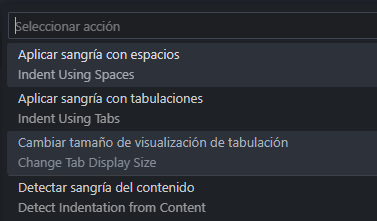
Cambiar el tamaño de tabulación en Visual Studio Code

Como un punto adicional, en visual studio code podemos cambiar el tamaño de la visualización de la tabulación, ten en cuenta que al pulsar la tecla TAB al momento de escribir código fuente en una línea se agrega un espaciado con un tamaño establecido.

Para cambiar el tamaño del espaciado, haz clic en la opción **Espacios**, el cual aparece en la esquina inferior derecha de la ventana de Visual Studio Code.

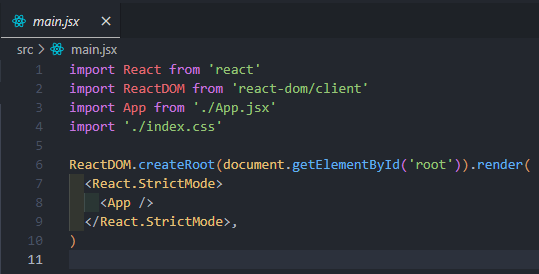


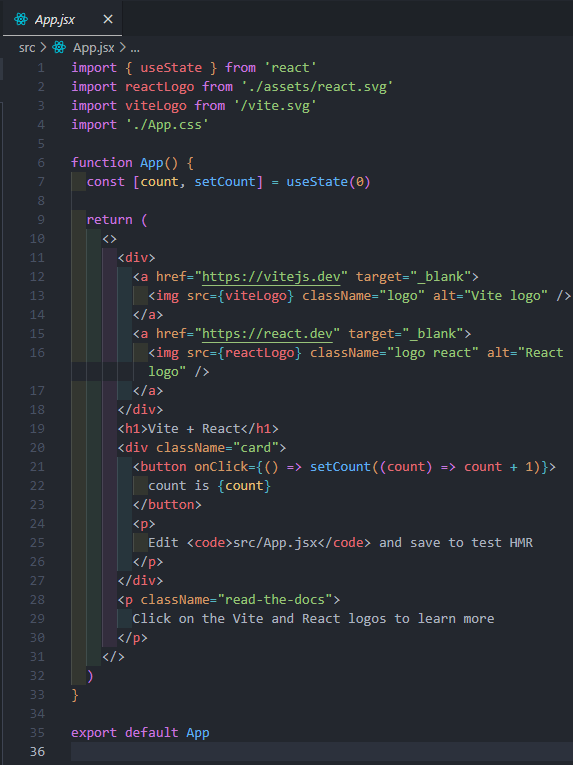
Cuando se abra el cuadro de diálogo en la parte superior de la ventana selecciona la opción **Cambiar tamaño de visualización de tabulación** y selecciona el valor **4** (es el valor por defecto para la mayoría de archivos). Realiza esta acción para todos los archivos que llevan la extensión **.jsx**.

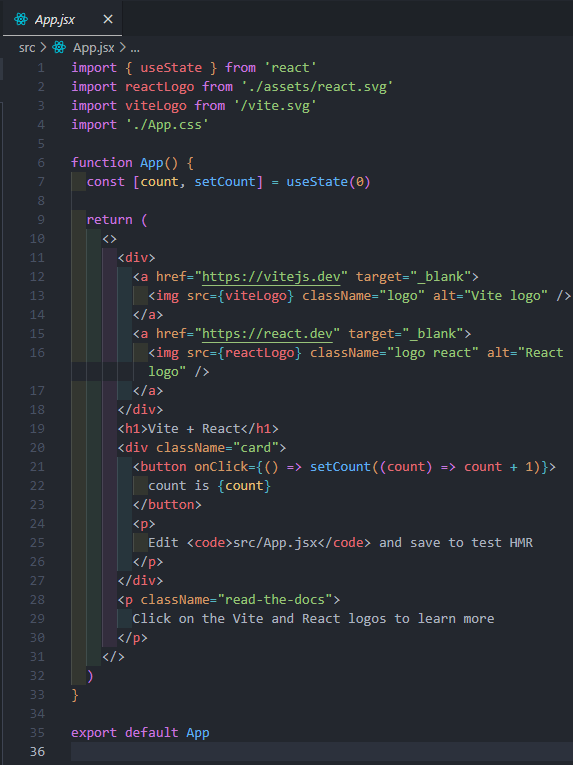


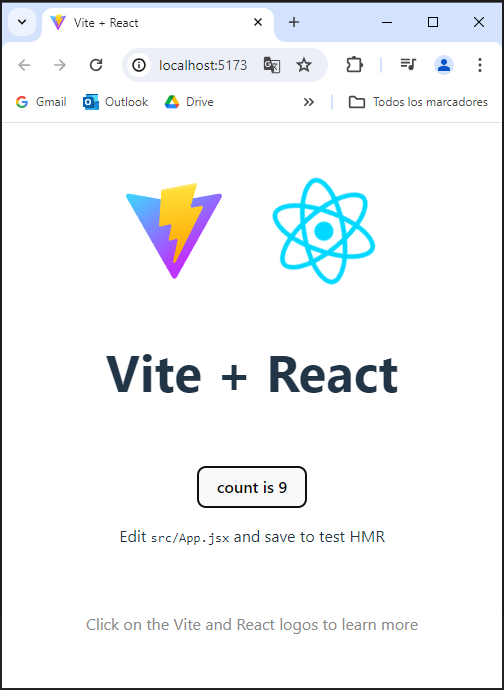
El proyecto generado por defecto

* El proyecto generado por defecto en Vite incluye un archivo **main.jsx** que carga el componente **App.jsx**.
* En **App.jsx**, se utiliza **useState** para crear un contador que incrementa cada vez que se pulsa un botón.
* El estado del contador se mantiene mientras la página está renderizada, pero se reinicia a 0 al actualizar la página.
* Más adelante veremos que para mantener el estado del contador entre actualizaciones de la página, se puede utilizar **useState** combinado con **SessionStorage** de HTML5 y JavaScript.









Uso de useState en React

Según la documentación de React, **useState** es una característica introducida en React 16.8 que permite manejar el estado y otras características de React sin necesidad de escribir clases.

Es útil para manejar el estado de una aplicación y también para gestionar el ciclo de vida de los componentes y eventos.

Ejemplo de contador con useState

Este código muestra un ejemplo básico de un contador implementado con **useState** en React. (obtenido de: <https://es.legacy.reactjs.org/docs/hooks-intro.html>).

import React, { useState } from 'react';

function Example() {

// Declara una nueva variable de estado, la cual llamaremos "count"

const [count, setCount] = useState(0);

return (

<div>

<p>You clicked {count} times</p>

<button onClick={() => setCount(count + 1)}>

Click me

</button>

</div>

);

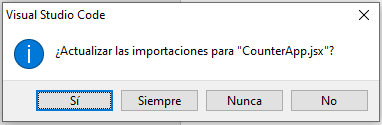
}

Empezar desde cero

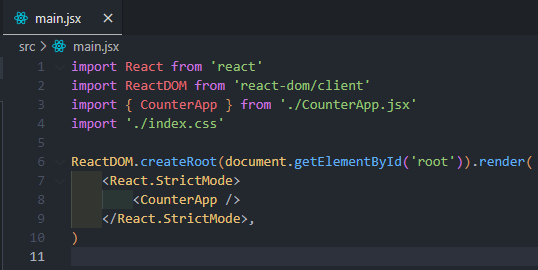
El objetivo de esta sección es empezar desde cero, por lo cual se va a omitir el proyecto generado por defecto y se va a modificar los componentes generados para implementar la funcionalidad del contador desde el principio.

Paso 1: El componente principal

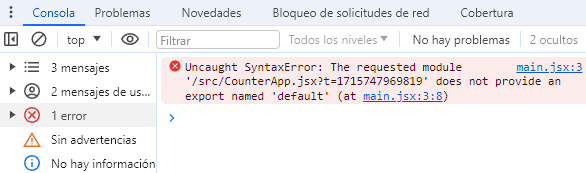
1. Cambia el nombre del archivo **App.jsx** a **CounterApp.jsx**. Si aparece una ventana para actualizar las importaciones, haz clic en **Sí**.



1. Borra todo el código contenido en el archivo **CounterApp.jsx**.
2. En el archivo **main.jsx**, importa el componente **CounterApp** desde **CounterApp.jsx** y luego renderizarlo dentro de la función **render** (ten en cuenta que el archivo **index.html** posee un elemento por defecto con el **id "root"**).

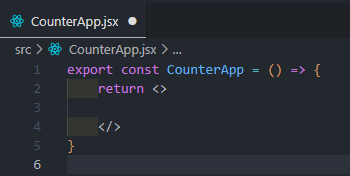


1. Si ejecutamos la aplicación, el navegador debe mostrar una pantalla en blanco y la consola debe mostrar un error. A plena vista indica que **CounterApp.jsx** no provee una función exportable por defecto.



Paso 2: Crear el componente CounterApp

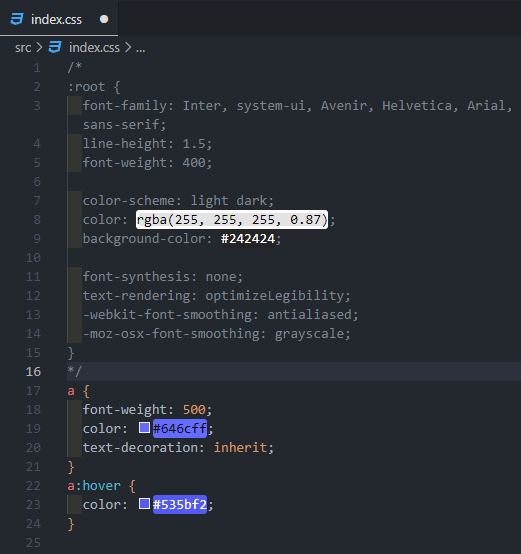
1. En el archivo **CounterApp.jsx**, crea el componente **CounterApp** como un functional component (un componente que incluye una función exportable) que retorna un fragmento sin contenido.

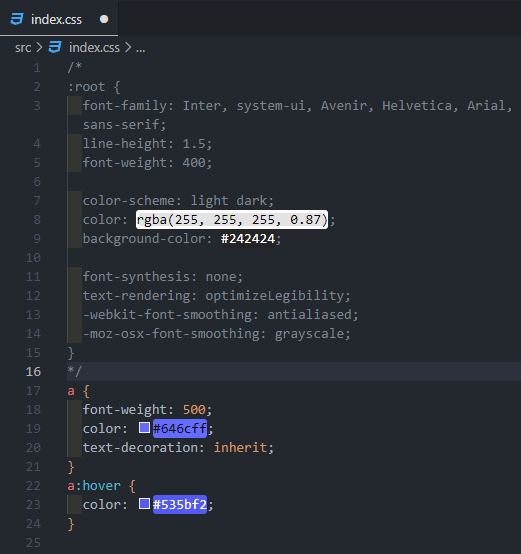


1. Esto solucionará el error que se mostraba en la consola de tal manera que la consola quede limpia, además en el navegador debe mostrar una página web en blanco.

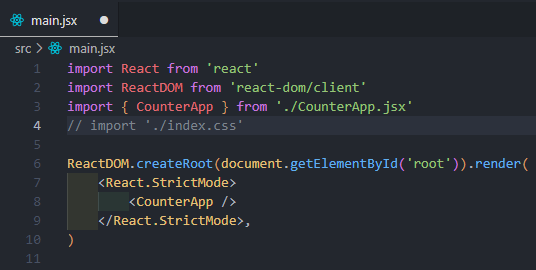
Paso 3: Eliminar estilos

1. En el archivo **index.css**, elimina los estilos para el elemento **root** (en CSS, el elemento **root** hace referencia a todos los estilos por defecto que se van a definir). En la siguiente imagen se ha comentado los estilos.





1. Elimina el archivo **App.css**. Es una buena práctica que cada componente tenga su propia hoja de estilos con el mismo nombre del componente y la extensión **.css**. Ten en cuenta que **index.css** es la hoja de estilos global, por lo cual se aplica a cada uno de los componentes.
2. Comenta la línea de código que llama a los estilos CSS en el archivo **main.jsx**.



1. La estructura de archivos del proyecto debe quedar de la siguiente manera:

