Creando proyecto InvoiceApp

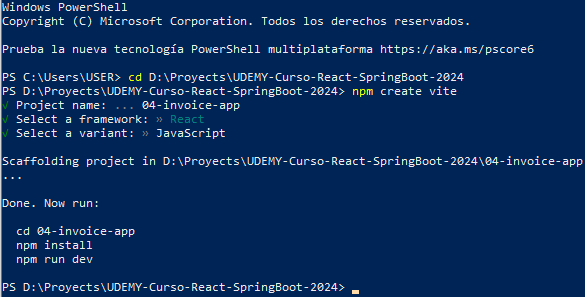
En esta sección se procederá a crear un nuevo proyecto desde la terminal de PowerShell, además de realizar las configuraciones iniciales al proyecto.

Pasos 1 al 5

Sigue estos pasos:

Paso 1: Comandos para crear un nuevo proyecto

La siguiente imagen muestra los comandos introducidos a una terminal de Powershell, el proyecto va a llevar el nombre de **"04-invoice-app"**.



Recuerda que el comando **cd** sirve para cambiar el directorio de trabajo y **npm create vite** se utiliza para crear un nuevo proyecto con Vite.

**Nota:** Puedes copiar la dirección de la carpeta especificada en la barra de direcciones del explorador de windows y con un clic derecho lo pegas en la consola.

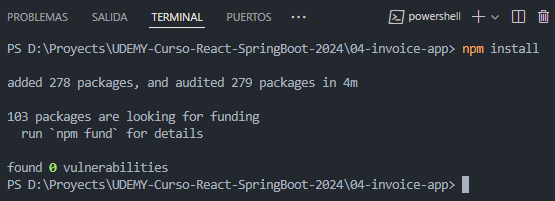
Paso 2: Configuraciones iniciales

Una vez creado el proyecto, se siguen estos pasos:

1. Abre Visual Studio Code.
2. Ve al menú **Archivo** y selecciona **Cerrar carpeta** para cerrar cualquier proyecto anterior.
3. Arrastra la carpeta **04-invoice-app** al editor de Visual Studio Code para abrir el proyecto.

Para generar el proyecto con las dependencias del **package.json** y crear la carpeta **node\_modules**, se realiza lo siguiente:

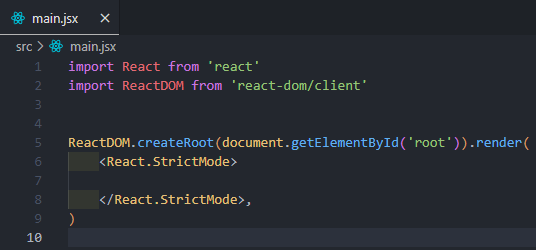
1. En Visual Studio Code, se abre una nueva terminal desde el menú **Terminal** y se elige la opción **Nuevo Terminal**. En caso de estar utilizando la misma terminal PowerShell, se introduce el comando: **cd 04-invoice-app** para ir a la carpeta raíz del proyecto.
2. También se puede utilizar el comando **ls** para listar los archivos, asimismo se debe mostrar un archivo llamado **packages.json** el cual contiene la lista de dependencias.
3. Se introduce el comando **npm install** para instalar y generar de forma automática la carpeta **node\_modules**.



Paso 3: Limpiar el proyecto

Se llevan a cabo las siguientes acciones:

1. Elimina los archivos **app.css** y **app.jsx**.
2. Borra el código fuente dentro del archivo **index.css**.
3. En el archivo **main.jsx**, elimina las importaciones hacia **index.css** y **app.jsx**, así como el componente **<App />**.
4. El archivo **main.jsx** debe quedar de la siguiente manera:

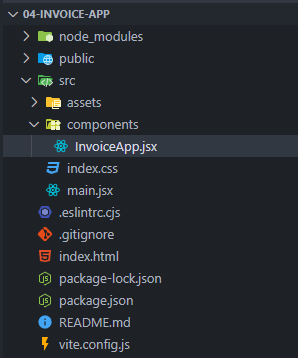


**Nota:** Recuerda que puedes cambiar la medida de la sangría cuando pulsas la tecla TAB, para realizar esto haz clic en el botón **espacios** y selecciona la opción **Cambiar tamaño de visualización de tabulación**, selecciona el valor por defecto **4**.

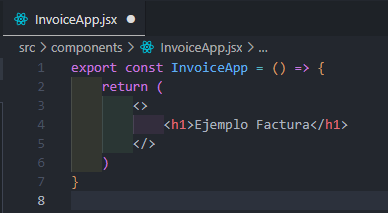


Paso 4: Crear el componente principal InvoiceApp

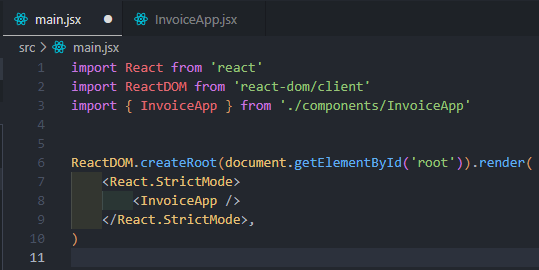
1. Crea una nueva carpeta (dentro de la carpeta raíz **src**) llamada **components**.
2. Dentro de la carpeta **components**, crea el archivo **InvoiceApp.jsx**.



1. Dentro del componente **InvoiceApp.jsx** define una función de flecha exportable con el mismo nombre del componente. La función va a retornar un fragmento que contiene un elemento **<h1>** con el texto **"Ejemplo Factura"**.



1. En el archivo **main.jsx**, importa el componente **InvoiceApp** desde su ubicación en **"./components/InvoiceApp"**. Se coloca el componente dentro del método **render**.

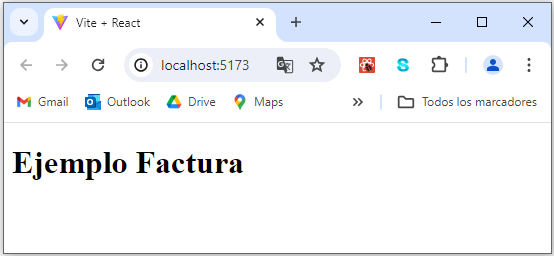


**Nota:** Es necesario tener abiertos ambos archivos (**main.jsx** e **InvoiceApp.jsx**) al mismo tiempo para poder importar de forma automática el componente al momento de escribir el nombre del componente en el código fuente.

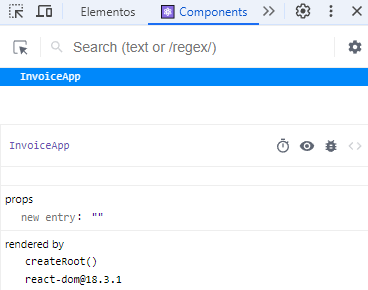
Paso 5: Levantar la aplicación

1. Se introduce el comando **npm run dev** en la consola para levantar la aplicación (recuerda guardar cambios en todos los archivos editados pulsando **CTRL + S**).
2. Para abrir la aplicación rápidamente, se mantiene pulsada la tecla **CTRL** y se hace clic en el enlace que aparece en la consola ([**http://localhost:5173/**](http://localhost:5173/)).



****

1. Para ver todos los componentes que integran la página web, pulsa la tecla F12 en el navegador para abrir la consola de Chrome. Ve a la sección **Components**.
2. Solamente se podrá ver el componente **InvoiceApp**, el cual no va a mostrar propiedades o props, porque aún no se han definido.



Hasta este momento se ha creado el esqueleto de la aplicación. Aún no se va a integrar con Spring Boot la parte del backend. Este ejemplo se centrará en React, manejando la data de la factura con todos sus elementos, utilizando hooks como **useState** y **useEffect** para manejar el estado y simular la carga de datos.

Más adelante, se modificará este ejemplo para conectarlo con un backend real utilizando Spring Boot y una base de datos.