**. CRA: Create-React-App**

**. VITE: Create Vite**

**1. CREAR UN PROYECTO CON create-react-app, necesitamos instalar:**

. a. Node.js (Entorno de ejecucion de JS)

. b. npm Node Package manager (Maneja las librerias de node)

**2. CONECCION DE ARCHIVOS A TRAVEZ DE root:**

**> public**

**> > index.html**

 <body>

    <div id="root"></div>

  </body>

**> src**

**> > index.js**

const root = ReactDOM.createRoot(document.getElementById('root'));

\*/

**VITE: Create Vite**

Es una plantilla de proyecto que puedes utilizar con NPM para crear rápidamente una aplicación web moderna basada en Vue.js o React

. npm create vite: Descargará y configurará automáticamente una estructura de proyecto básica para ti

**1. INSTALACION:**

1. **cd /Desktop/Frontend\_3/03\_Clase/Clase3**

2. **npm create vite**

3. √ Project name: **c5-vite-project**

4. √ Select a framework: **» React**

5. √ Select a variant: » **JavaScript**

**# LIBRERIAS FALTANTES**

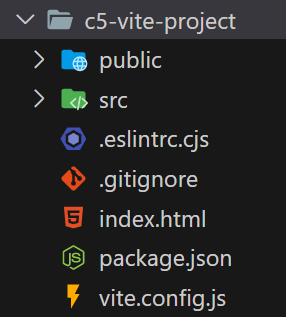
**> node\_modules**

**> package-lock.json**

**> package.json**

**dependencias:** Librerias propias de Reack

**Devdependencias:** Librerias agregadas por el desarrollador



**# INSTALAR LAS LIBRERIAS FALTANTES**

**6. cd c5-vite-project**

**7. npm i / npm install** - Instala o actualiza las librerias

**8. npm run dev**

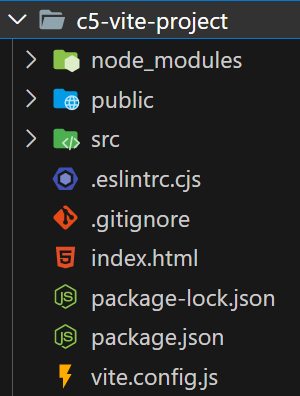
VITE v4.3.9 ready in 1133 ms

➜ Local: <http://localhost:5173/> - Terminal: o (Abre) / u (Cierra) el Localhost

➜ Network: use --host to expose

➜ press h to show help

**ARCHIVOS Y DIRECTORIO DE VITE**



**node\_modules:** Contiene todas las bibliotecas y paquetes de software de terceros que tu proyecto necesita para funcionar.

public: React contiene los archivos estáticos que se utilizarán al construir y ejecutar tu aplicación

favicon.ico

index.html: Este archivo es el punto de entrada de tu aplicación React

logo192.png

logo512.png

manifest.json

robots.txt

**src:** Contiene la mayor parte del código fuente de tu aplicación. Aquí se encuentran los componentes de React, Archivos JavaScript, Archivos CSS o estilos, archivos de imagen y otros recursos necesarios tu aplicación

App.css

App.js

App.test.js

index.js

logo.svg

reportWebVitals.js

setupTests.js

**.gitignore:** Especifica los archivos y directorios que deben ser ignorados y no deben ser rastreados por el control de versiones

**package-lock.json:**

Este archivo se utiliza para asegurar que todas las personas que trabajen en el proyecto tengan las mismas versiones de las dependencias instaladas, evitando problemas de compatibilidad.

Se actualiza automáticamente cada vez que se agregan, eliminan o actualizan las dependencias del proyecto garantizando que todos trabajen con las mismas versiones de dependencia

Contiene informacion detallada sobre las dependencia que el proyecto necesita

name:

version

dependencias

arbol de dependencia y como estas se relacionan

**package.json:**

Contiene la informacion del proyecto como dependencias requeridas, los comandos personalizados y la configuración específica

Nombre del proyecto

Version del proyecto

Dependencia

Script: Permite definir comandos personalizados en el archivo package.json que te permiten ejecutar. Ej: ejecutar pruebas

Autor y licencia

Configuración de webpack

**README.md:** archivo de texto que contiene información y documentación sobre tu proyecto.

Proporciona una descripción del proyecto, instrucciones de instalación, detalles sobre el uso y ejemplos, información sobre la colaboración y la licencia.

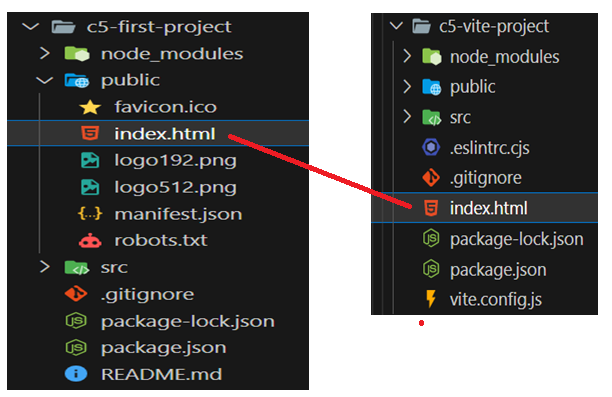
Recurso para que desarrolladores comprendan y utilicen tu proyecto de Reac

**3. DIFERENCIAS ENTRE CRA Y VITE**

**Diferencia N°1**

**CRA:** c5-vite-project/public/index.html

**VITE:** c5-vite-project/index.html: // Aparece al mismo nivel que public, src, package.json etc.

****

**Diferencia N°2**

**CRA: package.json**

    "@testing-library/jest-dom": "^5.16.5",

    "@testing-library/react": "^13.4.0",

    "@testing-library/user-event": "^13.5.0",

**VITE: package.json**

{

// Tenemos que configurar el testing

}

**Diferencia N°3**

**CRA: Index.html** (Mas lineas de codigo)

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

  </body>

</html>

**VITE: Index.html** (Menos lineas de codigo)

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <link rel="icon" type="image/svg+xml" href="/vite.svg" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Vite + React</title>

  </head>

  <body>

    <div id="root"></div>

    <script type="module" src="/src/main.jsx"></script> <!--Reemplaza al index.js

    En CRA, los archivos .js o .jsx los reconoce igual

**En VIRE Si devolvemos elementos html o componetnes tiene que ser .jsx**

    -->

  </body>

</html>

**Diferencia N°4**

**CRA: Index.js**

const root = ReactDOM.createRoot(document.getElementById('root')); // Captura root en una variable

root.render( // rederiza la variable

);

**VITE: main.jsx**

ReactDOM.createRoot(document.getElementById('root')).render( // Captura y renderiza en una linea

)

**Diferencia N°5**

**CRA: App.js**

function App() {

  return (

    <div className="App"> // Se utiliza className ya que class es una palabra reservada de JS

Codigo…

    </div>

**VITE: App.jsx**

function App() {

  const [count, setCount] = useState(0)

  return (

      <> // Fragment: Etiqueta padre

      Codigo…

    </>

  )

**Diferencia N°6**

**CRA:** 260Mb

**VITE:** 58,9 Mb (Mas liviano)

**# HERRAMIENTAS QUE VAMOS A UTILIZAR**

1. Webpack
2. EsLint
3. Jest
4. @testing-library
5. Fragment
6. Children

**a. Wenpack:** Empaquetador de modulos.

Convierte todos los modulos en un archivo estatico distribuible y comprimido legible para el navegador web

/\* .js .jpg .css .png .jpg .css .js .png

png .jpg .css .js .png .js .jpg .css

.js .jpg .css .png .jpg .css .js .png

png .jpg .css .js .png .js .jpg .css \*/

Wenpack= [.js .jpg .css .png]

**b. ESlint:**

. Optimiza el codigo,

. Mejora la sintaxis

. Advierte de patrones que pueden lanzar errores

. Configurar para generar convenio entre los desarrolladores. Ej: Los string van con comillas simples



**c.d. Jest y @testing-library:** Librerias para testear codigo

. En CRA se utiliza solo Jest. Comando: npm test: (Testea los archivos terminados en .test.js)

. Vite trabaja con una libreria llamada vitest

**e. Fragment <> </>**

Se utiliza para contener elemento que retornamos para no generar nodos extras en el DOM

Al no existir para el DOM, no aparaece como un elemento en (Navegador/inspeccionar/Elementos)

Para darle estilos al contenedor fragment <></> se debe convertir en un <div className="estilo"></div>;

return( // Debe retornar un solo elemento, en este caso un contenedor de elemetnos llamado fragment

<>

<div></div>

<p></p>

<h1></h1>

</>

)

**f. Children**

Se utiliza para retornar un elemento sin saber cual es el elemento contentedor del componente donde trabajamos

Children: Elemento que esta dentro de otro elemento padre