

Documentación ejercicio HTTP

Primero creamos nuestro Proyecto

En la parte de index colocamos todos los enlaces donde se redijeran a las paginas para la api y para la creación del pagina web

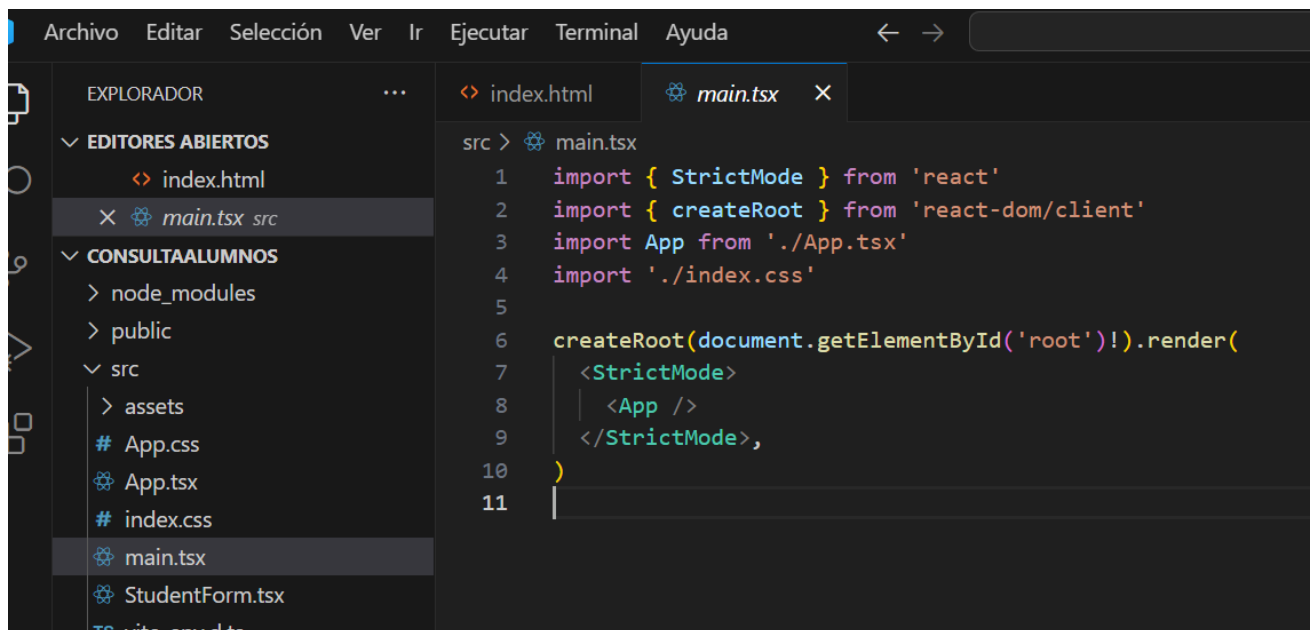
```
<> index.html > ...
1  <!doctype html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Consulta de alumnos</title>
7      <link rel="stylesheet" href="index.css"> <!-- Enlaza tu archivo CSS -->
8    </head>
9    <body>
10     <div id="root"></div>
11     
12     <script type="module" src="/src/main.tsx"></script>
13   </body>
14 </html>
15
```

StrictMode: Importa un contexto que activa comprobaciones adicionales y advertencias durante el desarrollo para ayudar a identificar posibles problemas en tu aplicación React.

createRoot: Importa una función de react-dom/client que se utiliza para crear un elemento raíz para la aplicación React dentro del DOM.

App: Importa el componente principal de tu aplicación React, que será renderizado en el elemento raíz.

'./index.css': Importa un archivo CSS llamado index.css que contiene los estilos para tu aplicación.



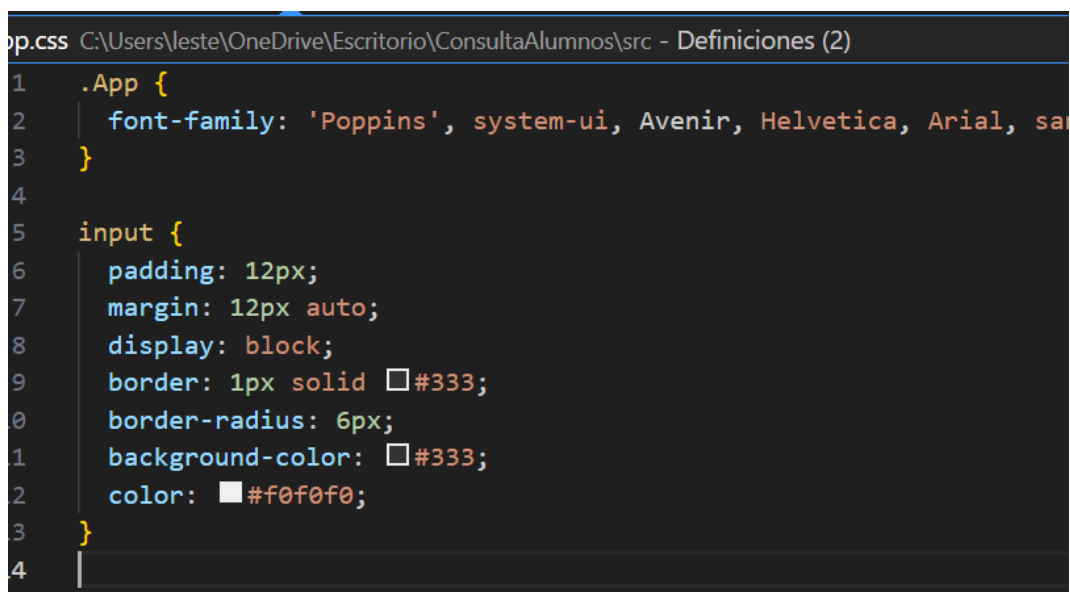
```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  <  >

EXPLORADOR  ...
  EDITORES ABIERTOS
    <> index.html
    ✕ main.tsx src
  CONSULTAALUMNOS
    > node_modules
    > public
    > src
      > assets
      # App.css
      # App.tsx
      # index.css
      # main.tsx
      # StudentForm.tsx
      TS vite-env.d.ts

src > main.tsx
1  import { StrictMode } from 'react'
2  import { createRoot } from 'react-dom/client'
3  import App from './App.tsx'
4  import './index.css'
5
6  createRoot(document.getElementById('root')!).render(
7    <StrictMode>
8      <App />
9    </StrictMode>,
10 )
11
```

En App.tsx establecemos la estructura básica de una aplicación React y definimos el componentes principal llamado App

App.css es solo para darle estilos



```
App.css  C:\Users\leste\OneDrive\Escritorio\ConsultaAlumnos\src - Definiciones (2)
1  .App {
2    font-family: 'Poppins', system-ui, Avenir, Helvetica, Arial, sans-serif;
3  }
4
5  input {
6    padding: 12px;
7    margin: 12px auto;
8    display: block;
9    border: 1px solid #333;
10   border-radius: 6px;
11   background-color: #333;
12   color: #f0f0f0;
13 }
14
```

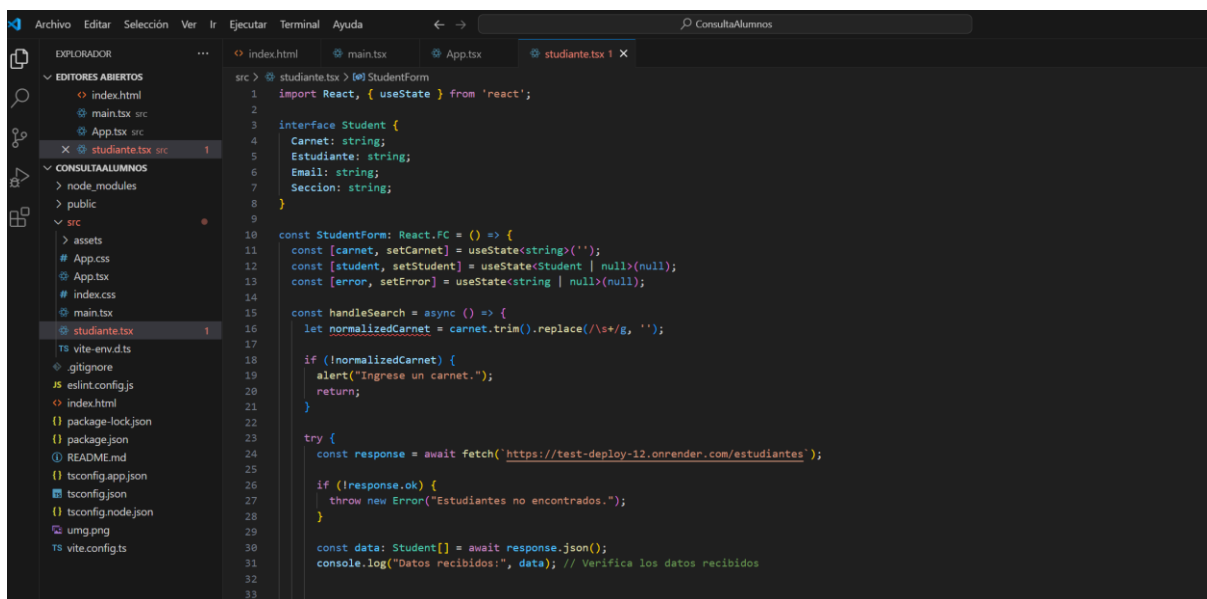
Luego el otro archivo `estudiantes` sirve para todo lo que vamos a ingresar y a buscarlo con el API

Este componente crea un formulario que permite a un usuario buscar a un estudiante por su carnet. Una vez que se ingresa el carnet y se presiona el botón de búsqueda, el componente realiza una petición a una API (en este caso, <https://test-deploy-12.onrender.com/estudiantes>) para obtener la información del estudiante correspondiente. Los datos del estudiante encontrado se muestran en los campos del formulario, y si no se encuentra, se muestra un mensaje de error.

Importaciones y Interfaz:

`import React, { useState } from 'react';` Importa las funcionalidades básicas de React y el hook `useState` que permite gestionar el estado de los componentes.

`interface Student:` Define una interfaz para representar un estudiante. Esta interfaz especifica las propiedades que tendrá un objeto de tipo estudiante (`carnet`, `nombre`, `correo electrónico` y `sección`).



```
src > estudiante.tsx > [0] StudentForm
1  import React, { useState } from 'react';
2
3  interface Student {
4    carnet: string;
5    estudiante: string;
6    email: string;
7    seccion: string;
8  }
9
10 const StudentForm: React.FC = () => {
11   const [carnet, setCarnet] = useState<string>('');
12   const [student, setStudent] = useState<Student | null>(null);
13   const [error, setError] = useState<string | null>(null);
14
15   const handleSearch = async () => {
16     let normalizedCarnet = carnet.trim().replace(/\s+/g, '');
17
18     if (!normalizedCarnet) {
19       alert("Ingresa un carnet.");
20       return;
21     }
22
23     try {
24       const response = await fetch('https://test-deploy-12.onrender.com/estudiantes');
25
26       if (!response.ok) {
27         throw new Error("Estudiantes no encontrados.");
28       }
29
30       const data: Student[] = await response.json();
31       console.log("Datos recibidos:", data); // Verifica los datos recibidos
32
33   }
```

Estado del Componente:

- **carnet:** Almacena el valor del carnet ingresado por el usuario.
- **student:** Almacena la información del estudiante encontrado (si existe).
- **error:** Almacena cualquier mensaje de error que ocurra durante la búsqueda.

Función `handleSearch`:

- Normaliza el carnet ingresado para eliminar espacios en blanco.
- Realiza una petición a la API utilizando `fetch`.
- Si la petición es exitosa, busca al estudiante en los datos recibidos.
- Actualiza el estado del componente con la información del estudiante encontrado o con un mensaje de error.

Funciones `handleClear` y `handleCancel`:

- Limpian los campos del formulario y restablecen el estado del componente.

Renderizado JSX:

- Crea el formulario con los siguientes elementos:
 - Un campo de entrada para el carnet.
 - Campos de texto deshabilitados para mostrar la información del estudiante (nombre, correo electrónico, sección).
 - Un botón para buscar.
 - Un botón para limpiar los campos.
 - Un botón para cancelar la operación.
 - Un mensaje de error si ocurre algún problema.

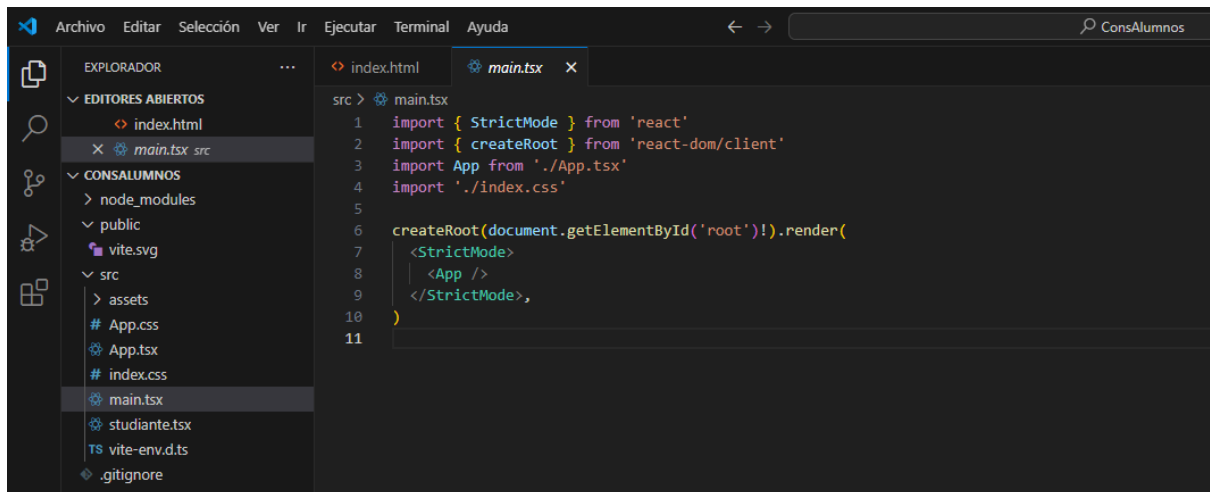
Luego tenemos el main.tsx

`import { StrictMode } from 'react':` Importa el componente `StrictMode` de la biblioteca React. Este componente activa comprobaciones adicionales y advertencias durante el desarrollo para ayudarte a identificar posibles problemas en tu aplicación.

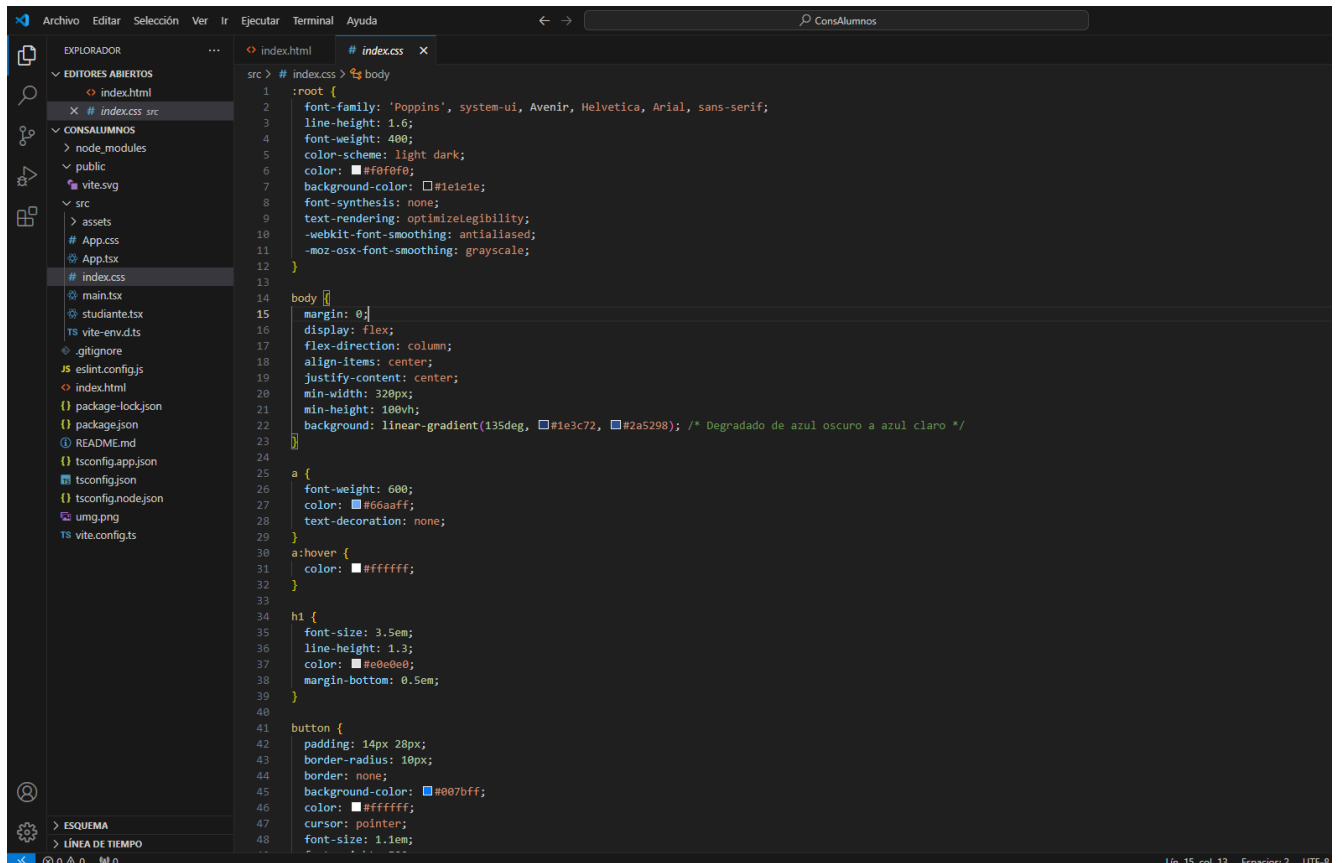
`import { createRoot } from 'react-dom/client':` Importa la función `createRoot` del módulo `react-dom/client`. Esta función se utiliza para crear un elemento raíz en el DOM donde se renderizará la aplicación React.

`import App from './App.tsx':` Importa el componente principal de tu aplicación, llamado `App`, desde el archivo `App.tsx`. Este componente contiene la lógica y la estructura de tu aplicación.

`import './index.css':` Importa el archivo CSS `index.css`, que contiene los estilos que se aplicarán a toda tu aplicación.



Diseño index css:



Este conjunto de reglas CSS define los estilos visuales de una página web, estableciendo una apariencia coherente y atractiva. Vamos a desglosar cada parte:

Estilos globales (:root)

- **Fuente principal:** Define la familia de fuentes principal para todos los elementos de la página, dando preferencia a 'Poppins' y otras fuentes comunes.
- **Altura de línea:** Establece un espacio entre líneas de texto para una mejor legibilidad.
- **Peso de la fuente:** Define el grosor de la fuente.
- **Esquema de color:** Permite que el navegador cambie automáticamente entre modos claro y oscuro según las preferencias del usuario.
- **Colores:** Establece los colores de fondo y texto por defecto, tanto para el modo oscuro como para el modo claro.
- **Otras propiedades:** Optimizan la representación de las fuentes en diferentes navegadores.

Estilo del cuerpo (body)

- **Márgenes:** Elimina los márgenes por defecto del cuerpo para un diseño más personalizado.
- **Flexbox:** Utiliza el modelo de diseño Flexbox para centrar el contenido de la página tanto horizontal como verticalmente.

- **Dimensiones mínimas:** Establece un ancho y alto mínimos para la página, asegurando que el contenido se vea bien en diferentes dispositivos.
- **Fondo:** Aplica un degradado de color para un efecto visual atractivo.

Estilos para enlaces (a), encabezados (h1), botones (button) y campos de entrada (input)

Estos estilos definen la apariencia de los elementos más comunes en una página web. Se personalizan el color, el tamaño de fuente, los bordes, los efectos de hover y otros aspectos visuales para crear una experiencia de usuario coherente.


Estilo para la imagen en la esquina (corner-image)

- **Posicionamiento:** Coloca la imagen en la esquina superior derecha de la página utilizando posicionamiento absoluto.
- **Tamaño:** Establece el ancho y alto de la imagen.
- **Bordes redondeados y sombra:** Aplica estilos opcionales para mejorar la apariencia de la imagen.

Media query para modo claro

- **@media (prefers-color-scheme: light):** Define estilos específicos para cuando el usuario prefiere el modo claro.
- **Cambios de color:** Ajusta los colores de fondo, texto y otros elementos para crear un tema más claro y legible.

Vista final



Consulta de alumnos

Carnet:

Nombres:

Correo Electrónico:

Sección: