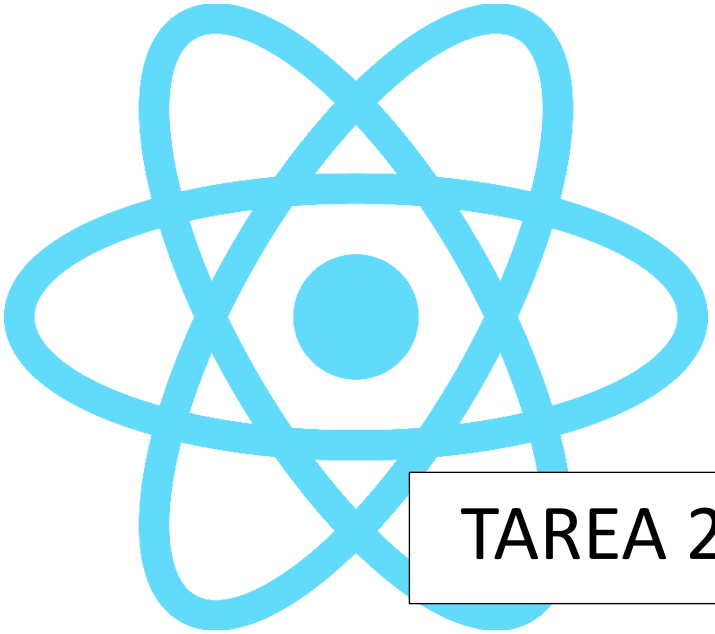


TAREA 2: REACT

Contenido

1) Ejecuta el proyecto

1. Abre una terminal en la carpeta my-react-app.

2. Ejecuta: npm install (si no lo hiciste).

3. Inicia el servidor: npm run dev

4. Abre la URL local que te muestre la terminal (por ejemplo: http://localhost:5173/).

2) Devuelve múltiples elementos con Fragment

5. Abre src/App.jsx y reemplaza el contenido por el ejemplo con dos encabezados.

6. Primero prueba con un envolviendo todo.

7. Luego cambia el por un Fragment vacío: <> ... </> y guarda.

```jsx
function App() {

  return (
    <div>
      <h1>Nivel 1 de React desbloqueado</h1>

    </div>
  );

}
```

El sirve simplemente para **envolver** varios elementos y que React no te dé error, ya que siempre debes devolver una sola pieza. El problema es que se queda ahí estorbando en el HTML final, aunque solo lo quieras para agrupar.

3) Renderizado: ¿dónde se “engancha” React?

8. Abre index.html y localiza: .

9. Abre src/main.jsx y localiza createRoot(...).render(...).

10. Escribe en tu informe (2-3 líneas) qué hace document.getElementById('root').

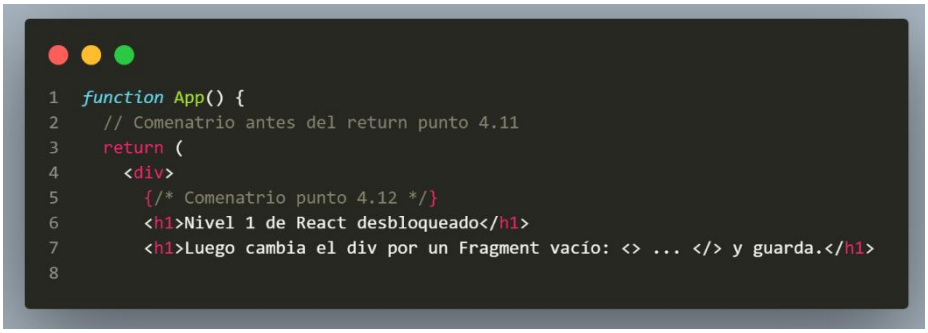
> Esta línea de código busca en el HTML el elemento que tiene el ID "root" para decirle a React exactamente dónde debe montarse y renderizar toda la aplicación. Es como marcar el punto de inicio en el index.html para que JavaScript sepa dónde empezar a dibujar los componentes.

4) Comentarios en React

11. Añade un comentario con // fuera del return.

12. Luego comenta una línea dentro del JSX usando: {/* ... */}.

13. Usa el atajo de VSCode: Ctrl+/ (Windows/Linux) o Cmd+/ (macOS).



```jsx
function App() {
  // Comenatrio antes del return punto 4.11
  return (
    <div>
      {/* Comenatrio punto 4.12 */}
      <h1>Nivel 1 de React desbloqueado</h1>
      <h1>Luego cambia el div por un Fragment vacío: <> ... </> y guarda.</h1>

```

En React existen dos formas de comentar según la ubicación:

- **Fuera del return:** Se usan comentarios de una sola línea con // como en JavaScript estándar.
- **Dentro del bloque JSX:** Es necesario envolver el comentario entre llaves y asteriscos {/* ... */} para que el motor de React lo reconozca correctamente como un comentario y no como texto plano.

5) Composición: varios componentes dentro de uno

14. Crea 3 componentes sencillos (UserComponent, ProfileComponent, FeedComponent).

15. Crea un ParentComponent que los renderice en orden.

16. Haz que App renderice <ParentComponent />.

```jsx
function App() {
  // Comenatrio antes del return punto 4.11
  return (
    <div>
      {/* Comenatrio punto 4.12 */}
      <h1>Nivel 1 de React desbloqueado</h1>
      <h1>Luego cambia el div por un Fragment vacío: <> ... </> y guarda.</h1>

      <ParentComponent/>
    </div>
  );

}
function ParentComponent() {
  return (
    <>
      <UserComponent />
      <ProfileComponent />
      <FeedComponent />
    </>
  );
}

function UserComponent() {
  return <h2>User component</h2>;
}

function ProfileComponent() {
  return <h2>Profile component</h2>;
}

function FeedComponent() {
  return <h2>Feed component</h2>;
}

export default App;
```

La composición consiste en crear una jerarquía donde un **componente padre** (como ParentComponent) contiene y renderiza a otros **componentes hijos** (UserComponent, ProfileComponent, etc.). Esto nos permite organizar la interfaz en piezas pequeñas, independientes y fáciles de mantener, que luego se juntan para formar una estructura completa.

Preguntas extra para el PDF

1. ¿Qué es un componente en React?

   Es una pieza de la interfaz independiente y reutilizable que funciona como un bloque de construcción para crear aplicaciones. Básicamente, es una función de JavaScript que define qué debe aparecer en la pantalla usando JSX.

2. ¿Por qué usamos Fragmentos (<>...</>) en lugar de un ?

   Los usamos para agrupar varios elementos sin meter un extra que no sirve para nada y que solo ensucia el HTML.

3. ¿Qué papel tienen index.html y main.jsx en el renderizado?

   El index.html es el archivo base que sirve como "esqueleto" y tiene el contenedor vacío donde se montará todo, mientras que el main.jsx es el punto de entrada de JavaScript que se encarga de inyectar nuestra aplicación de React dentro de ese archivo.

4. Explica con tus palabras qué significa “componer componentes”.

   Componer componentes es básicamente armar la interfaz como si fueran piezas de LEGO: vas metiendo componentes pequeños dentro de otros más grandes para crear estructuras más complejas.