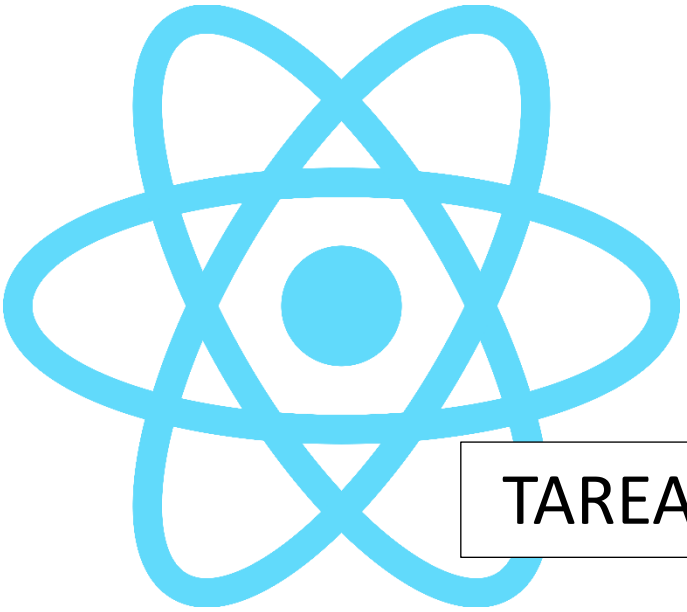

TAREA 5: REACT

Contenido

Parte A: Condicional con if + return

```
REACT > nivel1_react > my-react-app > src > App.jsx > App
1   function App() {
2     const user   = false;
3
4     if (user) {
5       return <button>Logout</button>;
6     }
7
8     return <button>Login</button>;
9   }
10
11  export default App;
```

Este código utiliza **retornos anticipados (return preventivo)** para decidir qué mostrar: si el usuario existe, el componente termina y devuelve el botón de Logout; si no, llega al final y devuelve el de Login.

Ilustración 1. Condiciona false. Elaboración propia

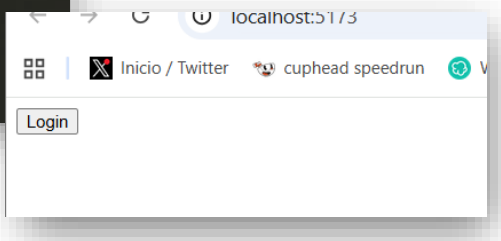

Ilustración 2. Visualización del resultado (Login). Elaboración propia

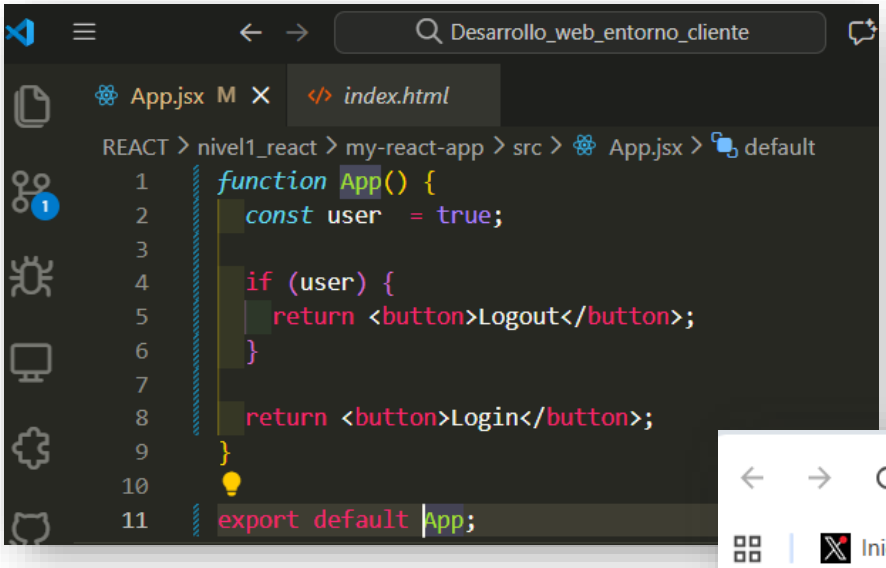

Ilustración 3. Condicional true. Evaluación propia

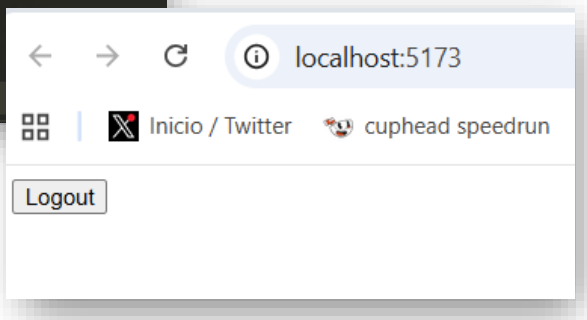

Ilustración 4. Visualización del resultado (Logout). Elaboración propia

Parte B: Renderizado parcial usando una variable

```jsx
function App() {
  const user  = true;

  let button = <button>Login</button>
  if (user) {
  button = <button>Logout</button>
  }
  return (
    <>
    <h1>Hola, esclavillo de DAW2</h1>
    {button}
    </>
  )
}

export default App;
```

Ilustración 3. Renderizado parcial. Elaboración propia

Este código utiliza una **variable de elemento** para decidir qué botón mostrar mediante un if antes de llegar al renderizado final.

Su función principal es **limpiar el return**, separando la lógica de decisión (Login/Logout) del HTML para que el componente sea más ordenado y fácil de leer.

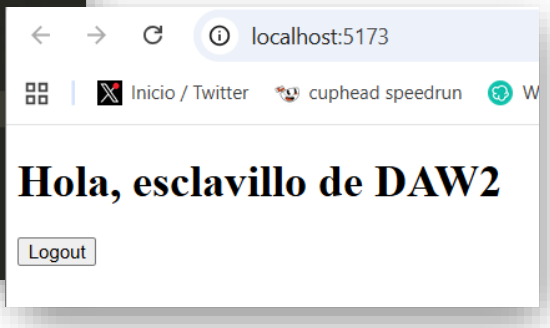

Ilustración 4. Visualización del resultado. Elaboración propia

Parte C: Renderizado en línea con el operador &&

```jsx
function App() {
  const newEmails = 2
    return (
      <>
      <h1>Hola chavules</h1>
      {newEmails > 0 && (
      <h2>Tienes {newEmails} nuevas palomas mensajeras
      esperando.</h2>
      )}
      </>
    )
}
export default App;
```

Ilustración 5. Renderizada línea. Elaboración propia

Este código utiliza el operador lógico && para mostrar un mensaje de alerta solo cuando hay correos pendientes. Si la variable newEmails es mayor a cero, el texto aparece en pantalla; de lo contrario, React lo ignora por completo y no muestra nada.

Ilustración 6. Visualización del resultado. Elaboración propia

2. Cambia newEmails a 0 y comprueba que el mensaje desaparece.

```jsx
function App() {
  const newEmails = 0
    return (
      <>
      <h1>Hola chavules</h1>
      {newEmails > 0 && (
      <h2>Tienes {newEmails} nuevas palomas mensajeras
      esperando.</h2>
      )}
      </>
    )
}
export default App;
```

Ilustración 7. newEmails. Elaboración propia

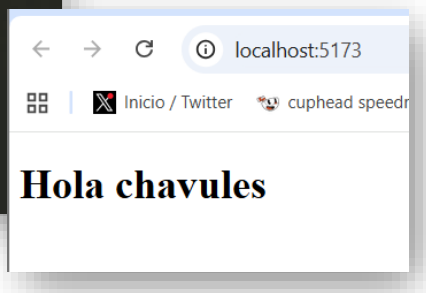

Ilustración 10. Visualización del resultado. Elaboración propia

Parte D: Renderizado en línea con operador ternario

```jsx
function App() {
const user = true
return (
<>
<h1>Hola chavules</h1>
{user ? <button>Logout</button> : <button>Login</button>}
</>
)
}
export default App;
```

Ilustración 11. Operador ternario. Elaboración propia

El código usa un **operador ternario** dentro del JSX para mostrar el botón de Logout o Login según si la variable user es verdadera o falsa. Es una forma compacta de renderizar contenido dinámico sin romper la estructura del HTML principal.

Ilustración 12. Visualización resultada. Elaboración propia

Miniretillo (pero de entrega obligatoria)

App.jsx M index.html

REACT > nivel1_react > my-react-app > src > App.jsx > ...

```jsx
import { useState } from "react"

export default function App() {

  const [user, setUser] = useState(null)
  const [newEmails, setNewEmails] = useState(0)

  const button = user
    ? <button onClick={() => setUser(null)}>Logout</button>
    : <button onClick={() => setUser({ name: 'Antoñico'
})}>Login</button>

  return (
    <>
    <h1>Nivel 5: Renderizado condicional</h1>
    {button}
    {user ? <p>Bienvenido, {user.name}.</p> : <p>Inicia sesión para
    continuar.</p>}
    <hr />
    <button onClick={() => setNewEmails((n) => n + 1)}>+1
    email</button>
    <button onClick={() => setNewEmails(0)}>Reset</button>
    {newEmails > 0 && (
    <h2>Tienes {newEmails} correos nuevos.</h2>
    )}
    </>
  )
}
```

Ilustración 13. Miniretillo. Elaboración propia

El código cambia según el estado del usuario: usa una **variable con un ternario** para alternar el botón de Login/Logout y el operador **&&** para mostrar notificaciones de correos solo cuando hay mensajes pendientes.

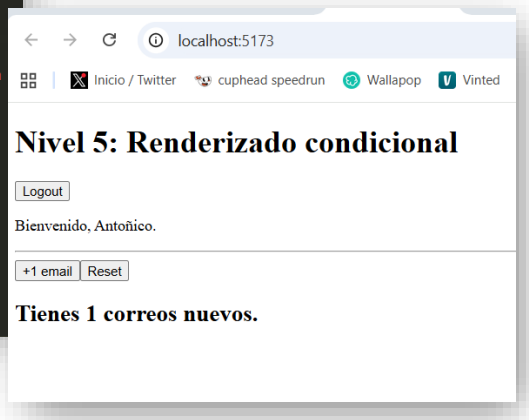

Ilustración 14. Visualización del resultado. Elaboración propia

Preguntas

1. **¿Cuál es la diferencia entre usar if/return y usar un ternario en JSX?**

   El if/return se usa fuera del JSX para elegir entre dos interfaces completas; el ternario se usa dentro del JSX para cambiar solo una parte específica del diseño.

2. **¿Cuándo te conviene usar && en lugar de un ternario?**

   Cuando solo tienes una condición "de verdad" (si se cumple, se muestra; si no, nada) y no necesitas una opción alternativa o "falsa".

3. **¿Qué ventaja tiene renderizar una parte dinámica en una variable (por ejemplo, button)?**

   Limpia el return principal de lógica compleja, haciendo que el código sea mucho más ordenado y fácil de leer al separar el "qué se muestra" del "cómo se muestra".

4. **Describe un caso real en una app donde el renderizado condicional sea imprescindible.**

   Un sistema de carga (Loading): mientras los datos bajan de internet, muestras un círculo girando; cuando llegan, muestras el contenido real.