TALLER REACT 1

NOMBRE ESTUDIANTE:

Ifigenia Becerra Soto
anybecerra1994@hotmail.com

DOCENTE:
Santiago Giraldo

SERVICIO NACIONAL DE APRENDIZAJE SENA
MUJERES DIGITALES G1
CAT VIRTUAL

DESARROLLO

1. EJERCICIO 1.

1.1 Investiga la historia de React y menciona dos hitos importantes en su desarrollo.

React es una biblioteca de JavaScript que fue creada por Facebook (ahora Meta) para facilitar el desarrollo de interfaces de usuario. Su historia comienza en 2011, cuando Jordan Walke, un ingeniero de software de Facebook, creó la primera versión de React como una solución interna para mejorar la experiencia de usuario de las aplicaciones de Facebook. Fue lanzada al público en 2013, y desde entonces ha revolucionado el desarrollo de aplicaciones web.

Dos hitos importantes en su desarrollo:

1. **Introducción de React Hooks (2019):** Los *React Hooks* fueron presentados en la versión 16.8 y cambiaron drásticamente la manera en que los desarrolladores manejan el estado y otros efectos en componentes funcionales. Antes de los hooks, el estado solo podía gestionarse en los componentes de clase, lo que limitaba la flexibilidad de los componentes funcionales. Con la introducción de hooks como `useState` y `useEffect`, React simplificó la escritura de código más limpio y reutilizable.
2. **Creación de React Fiber (2017):** React Fiber fue una reescritura completa del algoritmo de renderizado de React. Esta actualización permitió a React manejar actualizaciones en la interfaz de manera más eficiente, dividiendo el trabajo en pequeñas tareas para evitar que la UI se bloquee. Fiber también mejoró la capacidad de React para realizar actualizaciones de manera gradual, lo que fue crucial para experiencias de usuario más fluidas en aplicaciones complejas.

1.2 Escribe una breve explicación sobre por qué Facebook decidió crear React.

- Facebook decidió crear React en respuesta a los problemas de rendimiento y complejidad que enfrentaban en el desarrollo de sus aplicaciones web. A medida que su plataforma crecía, también lo hacían las demandas sobre la interfaz de usuario. El equipo de ingeniería se dio cuenta de que las actualizaciones de la interfaz eran lentas y difíciles de manejar con las herramientas existentes. React nació como una solución para crear interfaces de usuario más rápidas, dinámicas y fáciles de mantener. Uno de los principales desafíos era que cada vez que el estado de una aplicación cambiaba, todo el árbol de la interfaz de usuario necesitaba actualizarse. Esto provocaba que el código fuera difícil de gestionar, especialmente en aplicaciones grandes como Facebook. Con React, Facebook introdujo un enfoque basado en componentes y el "Virtual DOM", lo que permite que las actualizaciones solo afecten a los elementos que realmente cambian, mejorando la eficiencia y el rendimiento de las aplicaciones web (Facebook, 2013).

2. EJERCICIO 2.

2.1 Menciona tres ventajas de usar React en el desarrollo de aplicaciones web.

1. Reutilización de componentes : React permite dividir la interfaz de usuario en componentes pequeños y reutilizables. Esto facilita la creación y mantenimiento de aplicaciones complejas, ya que los desarrolladores pueden usar los mismos componentes en diferentes partes de la aplicación, ahorrando tiempo y esfuerzo (Facebook, 2013).
2. Actualizaciones eficientes : Gracias al *Virtual DOM* , React actualiza solo las partes de la interfaz que han cambiado, en lugar de volver a renderizar toda la página. Esto mejora considerablemente el rendimiento de la aplicación, haciendo que las interacciones sean más rápidas y fluidas (McDonald & Walke, 2017).
3. Gran comunidad y ecosistema : Desde su lanzamiento, React ha ganado una enorme comunidad de desarrolladores, lo que significa que hay una gran cantidad de recursos, bibliotecas y herramientas disponibles. Esto facilita resolver problemas, aprender nuevas técnicas y encontrar soluciones para casi cualquier desafío en el desarrollo web (Abramov & Clark, 2019).

2.2 Explica cómo el Virtual DOM mejora el rendimiento de una aplicación.

El *Virtual DOM* es una representación ligera de la interfaz de usuario que React mantiene en memoria. Cuando ocurre un cambio en el estado de la aplicación, React crea una nueva versión del *Virtual DOM* y lo compara con la versión anterior. Este proceso de comparación se llama *reconciliación*. Después de identificar las diferencias entre ambas versiones, React actualiza solo los elementos del verdadero DOM que han cambiado. Este enfoque es mucho más eficiente que volver a renderizar toda la página cada vez que se realiza una modificación, lo que mejora el rendimiento de la aplicación al reducir la cantidad de operaciones que el navegador tiene que hacer (McDonald & Walke, 2017).

3. EJERCICIO 3.

3.1 Define qué es una Single Page Application (SPA).

Una *Single Page Application* (SPA) es un tipo de aplicación web que carga una única página HTML al inicio y luego actualiza dinámicamente el contenido conforme interactúas con ella, sin tener que recargar la página completa cada vez que navegas. En lugar de hacer múltiples peticiones al servidor para cada nueva página, las SPA usan JavaScript para modificar solo las partes de la página que cambian, lo que mejora la experiencia de usuario al hacer que la navegación sea más rápida y fluida (Mozilla, 2021).

3.2 Explica cómo React facilita la creación de una SPA. Proporciona un ejemplo de cómo un componente de React puede actualizar la interfaz sin recargar la página.

React es una herramienta ideal para crear SPA porque está diseñada para construir interfaces de usuario dinámicas y reactivas. Gracias a su arquitectura basada en componentes y al uso del *Virtual DOM*, React permite que solo se actualicen las partes de la página que han cambiado,

sin necesidad de recargar todo el sitio. Esto se alinea perfectamente con el funcionamiento de una SPA, donde el contenido se actualiza dinámicamente en lugar de hacer una carga completa del sitio cada vez que se intera

```
export default App;
function ListaDeTareas() {
  const [tareas, setTareas] = React.useState(["Comprar pan", "Estudiar React"]);

  const agregarTarea = () => {
    setTareas([...tareas, "Nueva tarea"]);
  };

  return (
    <div>
      <ul>
        {tareas.map((tarea, index) => (
          <li key={index}>{tarea}</li>
        ))}
      </ul>
      <button onClick={agregarTarea}>Agregar tarea</button>
    </div>
  );
}
```

5. EJERCICIO 5.

5.1 Explica brevemente el propósito de las carpetas src y public en un proyecto React.

En un proyecto de React, las carpetas **`src`** y **`public`** cumplen con propósitos diferentes, pero ambos son fundamentales para que la aplicación funcione correctamente.

Carpeta `src`:
La carpeta `src` (source) es donde resides todo el código que los desarrolladores crean para construir la aplicación. Aquí es donde se encuentran los componentes de React, las hojas de estilo, los archivos JavaScript, y cualquier otra lógica que da vida a la app. En resumen, es el corazón de la aplicación, ya que React toma el contenido de esta carpeta y lo convierte en una interfaz interactiva para los usuarios (Facebook, 2013).

Carpeta `public`:
La carpeta `public`, por otro lado, contiene los archivos estáticos de la aplicación. Esto incluye el archivo `index.html`, donde React monta toda la aplicación, así como cualquier imagen, fuente o archivo que no va a cambiar durante el tiempo de ejecución. Los archivos en esta carpeta se entregan tal cual al navegador del usuario. Por ejemplo, cuando se carga la app, el navegador toma el `index.html` de esta carpeta y React luego lo utiliza como base para renderizar el resto de la aplicación (Facebook, 2013).

6. EJERCICIO 6.

6.1 Explica cómo JSX se diferencia del HTML tradicional.

JSX (JavaScript XML) es una extensión de JavaScript que, aunque se parece mucho al HTML tradicional, tiene varias diferencias importantes que lo hacen más flexible y funcional dentro de React. A simple vista, pareciera que estás escribiendo HTML, pero en realidad estás trabajando con una sintaxis que combina JavaScript y elementos de la interfaz.

Una de las principales diferencias es que, en JSX, puedes insertar directamente código JavaScript dentro de las etiquetas utilizando llaves {}. Esto significa que puedes pasar variables, ejecutar funciones o realizar cálculos en medio del marcado, algo que no es posible en HTML tradicional. Otra diferencia notable es la forma en que JSX maneja los atributos de las etiquetas. En lugar de utilizar class como en HTML, en JSX debes usar className, porque la palabra class ya está reservada en JavaScript. También hay otros atributos que cambian, como for, que en JSX se convierte en htmlFor. Además, en JSX es obligatorio que todas las etiquetas estén correctamente cerradas, incluso aquellas que en HTML no requieren cierre, como <img> o <input>. Finalmente, aunque JSX se parece al HTML, no es lo mismo. JSX se convierte en código JavaScript puro que React utiliza para crear los elementos de la interfaz de usuario. Bajo el capó, React transforma este código JSX en una serie de llamadas a funciones de JavaScript que, en última instancia, generan el contenido visible en la pantalla. Esto le da a JSX una ventaja sobre HTML al permitir que se construyan componentes dinámicos y reutilizables de manera más eficiente dentro de las aplicaciones React.

7. EJERCICIO 7.

7.1 Define los roles principales en un equipo SCRUM.

En un equipo Scrum, hay tres roles fundamentales que ayudan a que el proceso de desarrollo sea eficiente y bien organizado:

Product Owner:
El Product Owner es la persona encargada de representar los intereses del cliente o los usuarios. Su principal responsabilidad es gestionar el *Product Backlog*, que es una lista priorizada de todo lo que se necesita en el producto. El Product Owner decide qué características se deben desarrollar primero, asegurándose de que el equipo trabaje en lo que más valor aporte al negocio (Schwaber & Sutherland, 2020).

Scrum Master:
El Scrum Master es como un facilitador para el equipo. Se asegura de que todos sigan las reglas de Scrum y ayuda a eliminar cualquier obstáculo que impida al equipo avanzar. Su papel no es el de un jefe, sino más bien el de un guía, ayudando al equipo a ser más eficiente y a autoorganizarse (Schwaber & Sutherland, 2020).

Development Team (Equipo de desarrollo):
Este grupo está compuesto por los profesionales que realmente crean el producto o el software. El equipo de desarrollo es autoorganizado, lo que significa que deciden por sí mismos cómo abordar y completar el trabajo. Su objetivo es entregar una versión funcional y de calidad del producto al final de cada sprint (Schwaber & Sutherland, 2020).

7.2 Explica qué es un sprint y cómo se planifica.

Un sprint es un ciclo de trabajo en Scrum que normalmente dura entre una y cuatro semanas. Durante este período, el equipo se enfoca en desarrollar una lista específica de tareas que han sido extraídas del *Product Backlog*. El objetivo del sprint es entregar un incremento funcional del producto, es decir, una versión que pueda ser revisada o utilizada al final del ciclo.

La planificación del sprint ocurre al inicio de cada sprint. En esta reunión, el Product Owner presenta al equipo las tareas más prioritarias y el equipo de desarrollo decide cuántas de ellas pueden completar durante el sprint. Durante la planificación, el equipo también define un objetivo claro que quieren alcanzar para ese sprint, conocido como el *Sprint Goal* (Schwaber & Sutherland, 2020). Este proceso asegura que el equipo tenga una visión clara de lo que deben lograr en el ciclo de trabajo.

REFERENCIAS

- Abramov, D., & Clark, D. (2019). *Introducing Hooks*. React Blog. Facebook Inc. https://reactjs.org/docs/hooks-intro.html
- Facebook. (2013). *React: A JavaScript library for building user interfaces*. https://reactjs.org
- McDonald, B., & Walke, J. (2017). **Inside Fiber: In-depth overview of the new reconciliation algorithm in React**. React Blog. Facebook Inc. https://reactjs.org/blog/2017/05/17/react-fiber.html
- React. (2021). *Introducing JSX*. React Documentation. Facebook Inc. https://reactjs.org/docs/introducing-jsx.html
- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. Scrum.org. https://scrumguides.org