Paradigma Reactivo

Andrés Felipe Bernal Molina - 1023624342

Natalia Bernal Gutiérrez - 1001419475

Santiago Palacio Cárdenas - 1045076775

Angelo Gallego López - 1107838887

Sarah Yauripoma Cano - 1013338862

Facultad de Ingeniería, Universidad de Antioquia
Técnicas de Programación y Laboratorio

Docentes: Diana Margot Lopez Herrera

Medellín, 25 de Septiembre

Paradigma React

Consideraciones y suposiciones realizadas

- Valores no negativos: Todos los valores de entrada (precio por hora, horas, número de personas, gastos adicionales, costos de infraestructura, viáticos y porcentaje de riesgo) deben ser mayores o iguales a 0. Si alguno de estos valores es negativo, se muestra una alerta.
- Porcentaje de riesgo: El porcentaje de riesgo no puede superar el 50%. Si se introduce un valor mayor, se muestra una alerta.
- Número de personas: El número de personas debe ser un número entero. Si no lo es, se muestra una alerta.
- Cálculos de costos:
    - Costo de esfuerzo: Se calcula multiplicando el precio por hora, el número de horas y el número de personas.
    - Costo total: Se calcula sumando el costo de esfuerzo, los gastos adicionales, los costos de infraestructura y los viáticos.
    - Costo total con riesgo: Se calcula añadiendo al costo total el porcentaje de riesgo aplicado sobre el costo total.
    - Retención en la fuente: Se calcula como el 11% del costo total con riesgo.
    - Reteica: Se calcula como el 1% de la retención en la fuente.
    - IVA: Se calcula como el 19% de la suma del costo total con riesgo, la retención en la fuente y la reteica.
    - Costo final: Se calcula sumando el costo total con riesgo, la retención en la fuente, la reteica y el IVA.
- Actualización de resultados: Los resultados se actualizan cada vez que cambian los valores de entrada.

React es una biblioteca de JavaScript para construir interfaces de usuario. Su paradigma principal se basa en los siguientes conceptos:

1. Componentes: React divide la interfaz de usuario en componentes reutilizables. Cada componente es una pieza independiente de la interfaz que puede manejar su propio estado y lógica.
2. Declarativo: En lugar de manipular el DOM directamente, React permite describir cómo debería verse la interfaz de usuario en un estado dado. React se encarga de actualizar el DOM para que coincida con esa descripción.
3. Unidireccional: Los datos fluyen en una sola dirección, desde los componentes padres a los componentes hijos. Esto facilita el seguimiento de cómo los datos cambian a lo largo de la aplicación.
4. Estado y Props: Los componentes pueden tener un estado interno (state) y recibir datos de otros componentes a través de propiedades (props).
5. JSX: React utiliza JSX, una extensión de JavaScript que permite escribir código similar a HTML dentro de JavaScript. JSX facilita la creación de componentes visuales.

DOM (Document Object Model)

El DOM es una representación estructurada de un documento HTML o XML. Permite a los lenguajes de programación manipular la estructura, estilo y contenido del documento. En términos simples, el DOM es una interfaz que permite a los desarrolladores interactuar con el contenido de una página web de manera programática.

Hooks en React

Los Hooks son una característica de React que permite usar el estado y otras características de React sin escribir una clase, estos han revolucionado la forma en que los desarrolladores escriben componentes funcionales.
Ejemplos de Hooks:

1. useState: Permite agregar estado local a un componente funcional.
2. useEffect: Permite realizar efectos secundarios en componentes funcionales, como suscribirse a datos o realizar operaciones de limpieza.

React con Vite

Vite es una herramienta de construcción (build tool) moderna que proporciona un entorno de desarrollo rápido y eficiente para aplicaciones web. Vite se destaca por su velocidad y simplicidad, y es especialmente útil para proyectos de React.

Ventajas de usar Vite con React:

1. Inicio Rápido: Vite utiliza un servidor de desarrollo basado en ES modules, lo que permite un inicio casi instantáneo del servidor de desarrollo, incluso en proyectos grandes.
2. Hot Module Replacement (HMR): Vite ofrece una experiencia de desarrollo fluida con HMR, lo que significa que los cambios en el código se reflejan instantáneamente en el navegador sin necesidad de recargar toda la página.
3. Optimización de Producción: Vite utiliza Rollup para la construcción de producción, lo que resulta en paquetes optimizados y eficientes.
4. Configuración Sencilla: Vite requiere menos configuración en comparación con otras herramientas como Webpack, lo que facilita el inicio de nuevos proyectos.

Cómo iniciar un proyecto de React con Vite

Paso 1: Descargar e Instalar Node.js

- **Descargar Node.js**:
    - Ve al sitio web oficial de Node.js: https://nodejs.org/
    - Descarga la versión LTS (Long Term Support) recomendada para la mayoría de los usuarios.
- **Instalar Node.js**:

    - Ejecuta el instalador descargado y sigue las instrucciones en pantalla.
    - Asegúrate de que la opción para agregar Node.js al PATH esté seleccionada.
- **Verificar la Instalación**:
    - Abre una terminal (Command Prompt o PowerShell en Windows).
    - Ejecuta los siguientes comandos para verificar que Node.js y npm (Node Package Manager) se instalaron correctamente:
    - `node -v`
    - `npm -v`
    - Deberías ver las versiones instaladas de Node.js y npm.

Paso 2: Crear un Proyecto de React con Vite

- **Crear el Proyecto**:
    - En la terminal, navega al directorio donde deseas crear tu proyecto.
    - Ejecuta el siguiente comando para crear un nuevo proyecto de Vite con React:
    - `npm create vite@latest my-react-app --template react`
- **Instalar Dependencias**:
    - Navega al directorio del proyecto recién creado:
    - `cd my-react-app`
    - Instala las dependencias necesarias:
    - `npm install`

Paso 3: Iniciar el Servidor de Desarrollo

- **Iniciar el Servidor**:
    - Ejecuta el siguiente comando para iniciar el servidor de desarrollo:
    - `npm run dev`

Aplicación del paradigma

En el programa se implementa el paradigma de React para crear una calculadora de costos de desarrollo de software. A continuación, se explica cómo se utiliza React para resolver este ejercicio:

1. Componentes Funcionales y Hooks

- Componente Funcional: App es un componente funcional de React. Los componentes funcionales son funciones de JavaScript que retornan elementos de React.
- Hooks: Se utilizan varios hooks de React para manejar el estado y los efectos secundarios.
    - useState: Se usa para definir y manejar el estado local del componente.

    - useEffect: Se usa para manejar efectos secundarios, en este caso, para realizar cálculos cuando cambian los valores de los estados.

2. Estados Locales

Se definen varios estados locales utilizando el hook useState:

- precioPorHora, horas, personas, gastosAdicionales, porcentajeRiesgo, costosInfraestructura, viaticos: Estos estados almacenan los valores de entrada del usuario.
- resultados: Este estado almacena los resultados de los cálculos.

3. Efectos Secundarios

El hook useEffect se utiliza para realizar los cálculos cada vez que cambian los valores de los estados de entrada:

- Validaciones: Se realizan validaciones para asegurar que los valores sean válidos (no negativos, porcentaje de riesgo no mayor a 50%, número de personas entero).
- Cálculos: Se calculan los costos de esfuerzo, total, con riesgo, retención, reteica, IVA y el costo final.
- Actualización del Estado: Se actualiza el estado de resultados con los valores calculados.

4. Renderizado

El método return del componente funcional App define el JSX que se renderiza en la interfaz de usuario:

- Formulario: Se renderiza un formulario con campos de entrada para cada uno de los estados de entrada. Cada campo de entrada tiene un evento onChange que actualiza el estado correspondiente.
- Resultados: Si resultados no es undefined, se renderiza una sección que muestra los resultados de los cálculos.

5. Estilos y Recursos

- CSS: Se importa un archivo CSS (App.css) para los estilos.
- Imagen: Se importa y muestra un logo (logo.png).

Flujo del Código

1. Importaciones: Se importan las dependencias necesarias (React, useState, useEffect), el archivo de estilos (App.css), y una imagen (logo.png).

2. Definición del Componente App:

- Se definen varios estados locales usando useState para manejar los valores de entrada y los resultados de los cálculos.
- Se utiliza useEffect para realizar validaciones y cálculos cada vez que cambian los valores de los estados.

3. Validaciones:

- Se verifica que todos los valores sean mayores o iguales a 0.
- Se asegura que el porcentaje de riesgo no supere el 50%.
- Se valida que el número de personas sea un número entero.

4. Cálculos:

- Se calculan varios costos basados en los valores de entrada.
- Se actualiza el estado resultados con los valores calculados.

5. Renderizado:

- Se renderiza un formulario con campos de entrada para los diferentes valores.
- Se muestran los resultados de los cálculos si resultados no es undefined.

Conceptos Clave

const:

- Es una palabra clave en JavaScript que se usa para declarar variables cuyo valor no puede ser reasignado en la misma “iteracion”.

!Number.isInteger(Number(personas)):

- Number(personas): Convierte el valor de personas a un número.
- Number.isInteger(...): Verifica si el valor es un número entero.
- !: Niega el resultado de Number.isInteger, es decir, si personas no es un número entero, la expresión completa será true.

value:

- Es una propiedad de los elementos de formulario en React que se usa para establecer el valor del campo de entrada.

onChange={(e) => set(e.target.value)}:

- onChange: Es un evento que se dispara cuando el valor de un campo de entrada cambia.

- (e) => set(e.target.value): Es una función de flecha que toma el evento e como argumento y actualiza el estado con el nuevo valor del campo de entrada (e.target.value).

resultados.costo.toFixed(2):

- resultados.costo: Accede a la propiedad costo del objeto resultados.
- .toFixed(2): Formatea el número a dos decimales.

export default App;:

- export default: Es una declaración que se usa para exportar el componente App como el valor predeterminado del módulo. Esto permite que el componente sea importado en otros archivos.

StrictMode: Importa el componente StrictMode de React. StrictMode es una herramienta para destacar problemas potenciales en la aplicación. No renderiza nada en la interfaz de usuario, pero activa advertencias y comprobaciones adicionales en el modo de desarrollo.

createRoot: Importa la función createRoot de react-dom/client. Esta función es utilizada para crear un punto de entrada para la aplicación React en el DOM.

App: Importa el componente principal App desde el archivo App.jsx. Este es el componente raíz de la aplicación.

index.css: Importa el archivo de estilos CSS index.css para aplicar estilos globales a la aplicación.

createRoot(document.getElementById('root')): Selecciona el elemento del DOM con el id root y crea un contenedor raíz para la aplicación React. Este contenedor raíz es donde se montará la aplicación React.

render: Llama al método render en el contenedor raíz creado. Este método toma un elemento React y lo renderiza en el contenedor raíz.

StrictMode: Envuelve el componente App en StrictMode para activar las comprobaciones adicionales y advertencias en el modo de desarrollo.

App: El componente principal de la aplicación que se renderiza dentro del contenedor raíz.

El objetivo de la aplicación es saber cuanto es el costo de desarrollar un software, a partir de la consideración de 5 aspectos básicos
Costos: Los costos incluyen el esfuero + viaticos+ infraestructura. El esfuerzo se refiere a la cantidad de personas y por ende de horas hombre se debe invertir en el trabajo. Para esto, se debe conocer el valor hora de la persona que se va a emplear y el tiempo total que invertira en el proyecto, desde el comienzo

hasta el fin. Estos costos incluye la infraestructura en la nube donde se desplegará la aplicación al igual que aplicaciones como Github, Swagger, Sonar entre otras.
Gastos: consiste en los elementos o servicios que se compran o pagan para el proyecto y que no se recuperan, o que son necesarios para el funcionamiento de la empresa durante la ejecución del proyecto. Papelería, servicios, internet, energía.
Riesgos: Porcentaje de dinero que se asigna para cubrir las contingencia de personal, Gastos, Tiempo, en caso de materializarse el riesgo, de lo contrario queda como ganancia. Los riesgos no deben superar el 50
Impuestos: Los impuestos que por ley deben pagarse. La retención en la fuente (11 por ciento) se calcula sobre Costo+Gastos+Ganancia+Riesgo. El reteica se calcula sobre la Retención en la fuente y sera de (1 por ciento) IVA, se calcula sobre el valor de Costo+Gasto+Riesgo+Retenciones y es del 19 por ciento