Taller de nivelación 2da entrega

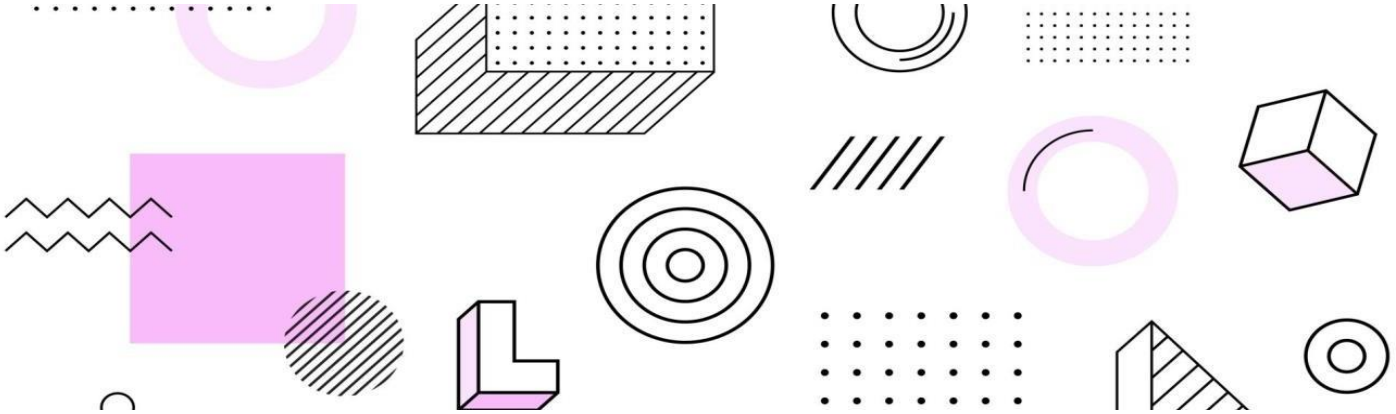
Presentado por:
Mannuel Fernando Granoble Pino

Presentado a:
Ing. Whitney Stevenson
Ing. Ana Ramírez

Makaia

Desarrollo web Frontend
Corte 6

Popayán,3 de Diciembre del 2023



MÓDULO SOBRE REACT JS

Preguntas teóricas

1. Explicar brevemente el concepto de ReactJS y sus principales características.
   RTA:

   ReactJS es una biblioteca de librerías, especializada en construcción de interfaces de usuario, sobre todo interactiva y reactivas.
   Dentro de sus principales características, destacan:
   - Su desarrollo por componentes→ estos componentes son pedazos de código reutilizables e independientes, para facilitar la programación sobre todo en proyectos de gran tamaño, dichos componente pueden ser muy grandes y complejos o muy pequeños e inservibles, esto depende del criterio del programador y su experiencia, lo ideal es que sean de un tamaño medio, y que se puedan reutilizar, en otras palabras imaginémonos un proyecto como un todo, pero ese todo es muy extenso entonces debemos dividirlo en componente para facilitar el desarrollo, pero al unirlos estos forman un todo llamado proyecto.
   - Estados y props → los estados son lo que utilizamos para gestionar los datos internos de un componente, claro está, que dicha gestión dependerá del criterio del programador y los props nos permiten pasar los datos de una componente a otro
   - Hooks →estos son funciones que nos permiten utilizar estados y otras características propias de componentes de clase en componentes funcionales, proporcionándole a estas características específicas que por sí solos no tenían.
   - Router → es una librería la cual permite facilitar la gestión y utilización de las rutas en una pagina de react, lo cual facilita la navegabilidad en el proyecto, permitiendo la renderización de un componente es especifico y no de toda una página entera, aportando a que nuestro proyecto sea una SPA.
   - Context → permite gestionar de manera más fácil, rápida y efectiva, la transmisión de datos de componentes a componentes, esto es muy útil sobre todo cuando se trata de proyectos grandes.
   - Virtual DOM → Virtual DOM es una representación jerárquica, así como el DOM, con la diferencia que este es una representación virtual y liviana de dicha estructura
   - JSX → es una extensión de JS, la cual nos permite mesclar en un componente código JS y HTML

2. Definir qué es un componente en ReactJS y mencionar los tipos de componentes que existen.
   RTA:

   Un componente es una pieza con un fragmento de código de la interfaz de usuario, que debe de ser reutilizables e independiente, la gran ventaja de estos es que permiten dividir un proyecto extenso en partes más pequeñas, y por ende mas manejables, facilitando de esa manera el desarrollo de los mimos, el mantenimiento y su gestionamiento, dichos componentes se pueden renderizar de manera independiente o combinarse y formar un todo, es decir interfaces complejas.
   Existen dos tipos de componentes:
   1. Componentes de clase: Estos son de mayor complejidad a la hora de programarlos, su sintaxis es más extensa, ya casi que no son utilizados, debido a que estos se trabajaban en las versiones iniciales de React, pero este avanzo y así mismo mejoro este tipo de cosas para lograr que la programación sea más amena y fácil, como dato importante cuando se declara un componente de clase, este debe extender de React.Component, para que pueda hacer uso de sus funciones.
   2. Componentes funcionales: Estos son más fáciles de usas, su sintaxis es más simple y es la versión mejorada que React les dio a los componentes de clase, en la actualidad estos son la forma más popular de programación con React, estos tienen la ventaja que reciben propiedades como argumentos y retornan elementos React.

3. ¿Qué es el Virtual DOM y cuál es su función en ReactJS?
   RTA:

   Virtual DOM es una representación jerárquica, así como el DOM, con la diferencia que este es una representación virtual y liviana de dicha estructura, la cual es manejada y gestionada por React en memoria, lo cual ayuda de manera significativa en el mejoramiento del rendimiento y la efectividad de nuestra aplicación.
   Su función básicamente es la de que, cuando se haga una actualización en la aplicación, este va a actualizar el virtual DOM primero, antes de hacerlos directamente en el DOM original, permitiendo mayor efectividad en el proceso y un mejor rendimiento a nivel general de la aplicación, en pocas palabras la optimización de las actualizaciones del DOM, reduce costos, reduce tiempos, mejora rendimiento y la efectividad de nuestro proyecto

4. ¿Qué es JSX en ReactJS y porqué es importante?
   RTA:
   JSX es una extensión JS que permite la mezcla de código tanto HTML como JS, es muy utilizado por ReactJS para el desarrollo y la creación de interfaces de usuario interactivas, gracias a que proporciona una sintaxis más clara y concisa para el desarrollo de elementos React.
   Trabajar con JSX es importante ya que optimiza el rendimiento de una aplicación, ya que su compilación se hace a través de JS las cuales en términos de rendimiento son mucho más eficientes a comparación de la compilación tradicional que se hace a través de la manipulación del DOM, también gracias a la fácil integración que tiene esta librería con React el desarrollo se vuelve más intuitivo, fácil y óptimo.
   Claro esta que no se puede dejar a un lado la legibilidad y la expresividad de JSX, ya que permite que el código creado en este, sea expresivo y fácil de comprender, puesto que se asemeja a la estructuración original del DOM que representa.

5. ¿Qué es un Hook en ReactJS y cuál es su propósito?
   RTA:

   Un Hook es una función especial la cual posibilita a los componentes funcionales teber acceso e interacción con diferentes características de React, tales como, los estados, el contexto, el efecto, entre otros.
   En esencia el propósito principal de los Hooks radica en facilitar la reutilización de la lógica de programación, posibilitando el intercambio de datos entre componentes, la gestión de los mimos y la simplificación la gestión de su ciclo de vida.

6. Mencionar al menos tres tipos de Hooks en ReactJS y explicar brevemente para qué se utilizan.
   RTA:

   1. useState → los estados son lo que utilizamos para gestionar los datos internos de un componente, claro está, que dicha gestión dependerá del criterio del programador.
   2. useContext →permite gestionar de manera más fácil, rápida y efectiva, la transmisión de datos de componentes a componentes, esto es muy útil sobre todo cuando se trata de proyectos grandes.
   3. useEffect → Facilita la ejecución de efectos secundarios en los componentes funcionales, dado que es utilizado en gran parte para hacer peticiones a la API o permite la renderización de su con tenido al inicio, cuando se ejecute el componente en el que esta albergado.

7. ¿Cuáles son las reglas de uso para los Hooks en ReactJS?
   RTA:
   1. Los Hooks solo se deben usar en los componentes funcionales, ya que estos les brindan a dichos componentes las propiedades o características de los componentes de clase, puesto que estos ya poseen su propio estado y ciclo de vida.
   2. Los Hooks solamente deben de ser llamados desde los componentes funcionales o Hooks personalizados.
   3. En caso de los Hooks personalizados deben iniciar con la palabra use, para que su identificación sea más fácil.
   4. Los Hooks no se deben usar dentro de bucles, condiciones o funciones anidadas, puesto que estos esta diseñados para ser tratados o utilizados en niveles superiores ya que el rastreo de los mismo es mucho más eficiente.

8. Explicar qué es React Router DOM versión 6, para qué se utiliza y cuáles son sus principales componentes y Hooks.
   RTA:

   Es una biblioteca que facilita la gestión de rutas en aplicaciones web construidas con React, permitiendo la presentación de determinados componentes según la URL, solicitada por el usuario en ese momento, lo cual ayuda enormemente en la creación de páginas SPA (Single Page Application).
   Esta Liberia se usa principalmente para la navegación sin la necesidad de recargar toda la página, gracias a que solamente renderizara los componentes necesarios por la URL y no todo el proyecto, también para el enrutamiento declarativo, es decir, la definición de como los componentes se deben mostrar según la URL solicitada por el usuario y para el

manejo del historial, a la hora de navegar por la página, puesto que el administrar ese historial, se le permitirá al usuario, acciones como las de retroceder y avanzar entre las distintas vistas del proyecto.

Los principales componentes que usa son:

- <BrowserRouter> </BrowserRouter> → Este componente es lo que le informa a React que todo lo que este contenido en él, serán rutas, las cuales nos permitirán realizar la navegabilidad dentro de la aplicación
- <Route></Route> → Este es un componente importante, puesto que es el que relaciona la tura con el componente a renderizar, estos pueden contener componentes anidados, para formar rutas secundarias
- <Link> </Link> Este componente nos genera el enlace directo a las diferentes rutas de la aplicación.

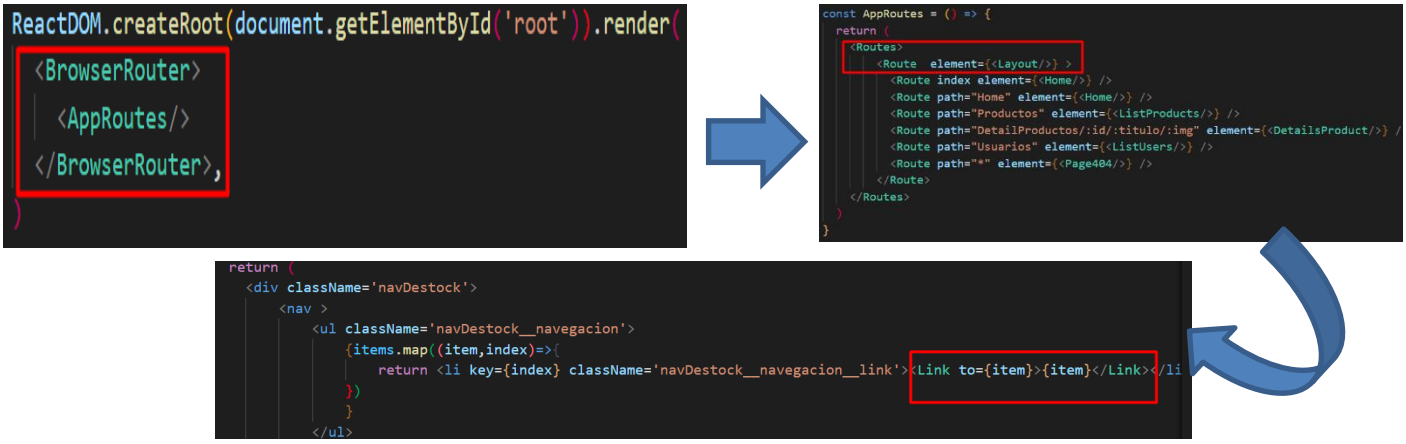
Los principales Hooks que usa son:

- useNavigate → es una función de navegación, la cual se usa para cambiar la ruta de la app
- useParams → Este nos permite acceder a los parámetros que enviamos a través de la ruta.

9. Explicar cómo se realiza la navegación entre diferentes páginas utilizando React Router DOM.
   RTA:

   1. Se debe instalar la librería a través del comando npm install react-router-dom
   2. Se deben definir las rutas a través de los componentes <BrowserRouter> </BrowserRouter> y <Route></Route>
   3. Se debe establecer los enlaces a las diferentes rutas con el componente <Link></Link>

   Ejemplo:



10. ¿Cómo se definen rutas en React Router DOM?
    RTA:

    Las rutas se definen a través de los componentes </BrowserRouter>, <Route></Route> y <Link></Link>, pero en especial a través de <Route></Route>, ya que este es el que se encarga de unir la URL con el componente que se mostrara en la pantalla una ves sea necesario renderizar dicha ruta

Ejemplo:

```
const AppRoutes = () => {
  return (
    <Routes>
        <Route  element={<Layout/>} >
          <Route index element={<Home/>} />
          <Route path="Home" element={<Home/>} />
          <Route path="Productos" element={<ListProducts/>} />
          <Route path="DetailProductos/:id/:titulo/:img" element={<DetailsProduct/>} />
          <Route path="Usuarios" element={<ListUsers/>} />
          <Route path="*" element={<Page404/>} />
        </Route>
    </Routes>
  )
}
```

NOTA: cabe aclarar que podemos crear rutas anidadas para general URLS secundarias o pasar parámetros en las mismas rutas, como se puede evidenciar en la imagen anterior.

11. Describir cómo crear un proyecto ReactJS con Vite

RTA:

1. Se debe abrir una terminal, en especial git bash here, en la ubicación donde desees almacenar tu proyecto.
2. Se debe colocar el siguiente comando npm créate vite@latest nombre_Proyecto -- --- template react
3. Algunas veces te da unas opciones, entonces se debe se debe elegir react y luego JavaScript, pero eso cuando te las pida, ya que no siempre ocurre.
4. Se debe abrir en el visual studio el proyecto que se creó en tu computador.
5. Se debe abrir una terminal de tu proyecto en visual Studio y colocar npm install, para que instale todas las dependencias que sean necesarias para correr el proyecto.
6. Luego en la terminal se coloca npm run dev, para verificar si tu proyecto fue creado con éxito. Luego de ejecutarse es comando te da un link, al cual al darle clic y si todo esta bien debe salirte esta pantalla.

7. Aquí ya se creo un proyecto con react y vite de manera exitosa, obviamente para trabajar debemos limpiar o eliminar las cosas que no nos sean útiles, pero eso ya depende del criterio de cada programador.

12. Describir cómo desplegar un JSON server en cualquier Hosting free o servicio en la nube gratuito de su elección

RTA:

A continuación, se describirá el paso a paso para el despliegue del Json server en render:

1. Se parte asumiendo de que ya se tiene creado un proyecto que contenga el JSON server que deseamos desplegar.

2. Subimos dicho proyecto a nuestra cuenta GitHub.

3. Se debe abrir en el navegador la página oficial de render (render.com). NOTA: esta debe ser inicializada con nuestra cuenta GitHub

4. Seleccionamos la opción que dice “DASHBOARD”

5. Seleccionamos la opción New y posteriormente el ítem de web Service.

6. Seleccionamos la opción Build and deploy from a Git repository y presionamos en botón “Next”.

7. Luego seleccionamos el repositorio que deseamos desplegar y presionamos el botón “Connect”

8. Luego le ponemos un nombre, no modificamos o agregamos ningún dato más y nos dirigimos a la parte inferior y presionamos el botón “Create Web Service”

9. Esperamos a que se ejecute y cuando en la pantalla negra salga el letrero que dice “is live”, quiere decir que ya esta desplegado nuestro JSON server.

10. En la parte superior en letras color azul e iniciando con https…… se encuentra la URL de nuestro despliegue lista para ser usada.

13. Describir cómo desplegar una aplicación en cualquier Hosting de su elección.
RTA:

A continuación, se explicará el despliegue de una aplicación en vercel:

1. Se parte de la idea de que ya se posee un proyecto formado.
2. Se sube el proyecto a un repositorio GitHub
3. Se ingresa a la página oficial de vercel (vercel.com)
4. Luego se selecciona el botón “Add New” y seleccionamos el ítem proyect.
5. Seleccionamos el repositorio que deseamos desplegar y le damos “Import”.
6. Sin modificar ni agregar ningún dato nos dirigimos a la parte inferior y presionamos el botón “Deploy”
7. Ya por último esperamos a que cargue, hasta que nos muestre una pantalla con la imagen de nuestro proyecto.
8. Presionamos en botón “Continue to Dashboard”
9. Por último, nos aparecerá la imagen del proyecto, pero al lado una información, de la cual, el primer dato, que se encuentra debajo de “Deployment ” será la URL del despliegue de nuestro proyecto.

MÓDULO SOBRE GESTION DE ESTADOS Y DATOS CON REACT CONTEXT Y USEREDUCER

Preguntas teóricas

1. ¿Qué es `React Context` y para qué se utiliza en el desarrollo web con React?
   RTA:

   React Context es una librería de React, la cual nos permite gestionar de manera más fácil, rápida y efectiva, la transmisión de datos, sin la necesidad de enviar explícitamente las props de componentes a componentes, esto es muy útil sobre todo cuando se trata de proyectos grandes y para evitar la propagación excesiva de props a través de múltiples niveles del proyecto.

2. Describir cómo se crea un contexto en React y cómo se proporciona y consume información a través de él.
   RTA:

   1. Se debe crear un contexto, gracias a la función 'createContext'
   2. Se debe envolver los componentes a los cuales le queremos transmitir la información, dentro del componente o nombre de la variable que contenga mi contexto seguido le agregamos .Provider value={} y dentro de las llaves le pasamos la información a ese contexto, dicha información podrá ser utilizada por la aplicación.
   3. Ya los componentes hijos del contexto que creamos podrían acceder a la información, pero esto se podrá hacer de dos manera, ambas validas, solo que es criterio del programador cual es la de su preferencia, la primera forma de acceder a la información de mi contexto es <nombreContexto.Consumer>, y lo único que debemos hacer es en el componente hijo del contexto que queremos acceder a la información, primero importamos el componente donde esta nuestro contexto , y luego dentro de la etiqueta que se nombró anteriormente, se desestructurara la información requerida:

      <nombreContexto.Consumer>
      {valor => {valor} }
      </nombreContexto.Consumer>

      Y la segunda manera para poder acceder es a través de useContext y para esto primero se debe importar la librería useContext de react, y luego se captura en una variable el contenido del contexto a través de la función useContext, a la cual se le deberá pasar como parámetro el nombre del contexto a capturar, en este caso cabe aclarar que se puede capturar toda la información del contexto se puede solamente desestructurar la información requerida

      Const valor = useContext(nombreContexto)

3. ¿Cuál es la ventaja de utilizar `React Context` en lugar de pasar `props` a través de múltiples componentes?
   RTA:

   Sin duda alguna una de las principales, es la simplificación y legibilidad de código,

puesto que, al no tener que realizar una propagación excesiva de props, el código se vuelve más limpio y entendible para los desarrolladores, evitando así que nuestro proyecto se torne engorroso y difuso al hacer una grana anidación de paso de parámetros, a través del proyecto.
Por otro lado la facilidad e transmisión de los datos puesto que en proyecto grandes que pasan de 3, 4, 5 o más niveles, el hacer esa anidación de props puede fomentar la generación de errores, esto también acompañado a que la escalabilidad es mejor puesto que si se llegara hacer un cambio, dicho cambio de debería modificar en un o máximo 2 ítems, pero si es por props, dicho cambio deberá modificarse en múltiples tiems del proyecto, claro esta que gracias a esto se vuelve más mantenible en el tiempo, mejorando así un poco la esperanza del ciclo del vida del software.

4. Explicar el propósito de `useReducer` en React y cómo se diferencia de `useState`.
   RTA:

   useReducer es un hook de react, el cual tiene como propósito gestionar estados más complejos y lógica de actualización avanzada en estructuras complejas como objetos o matrices en componentes, a diferencia de useState el cual se encarga de gestionar solamente un único estado.
   Además 'useReducer' permite definir una función reductora, la cual tiene como objetivo de centralizar la lógica de actualización del estado, proporcionando así una solución más organizada para casos de actualizaciones más avanzadas.
   En pocas palabras 'useReducer' es útil sobre todo en los casos cuando la lógica de actualización de un estado se vuelve más compleja.

5. Describe los argumentos que toma la función `useReducer`.
   RTA:
   - 'reducer'→ Es una función suyo propósito es el de especificar la manera en que el estado deber ser actualizado en respuesta a una acción, dicha función recibe 2 parámetros como argumentos, los cuales son: 'state' que representa el estado actual y 'action' el cual representa la acción, la cual describe el tipo de actualización que se debe realizar en el estado, dicha función retorna un nuevo estado, dependiendo del tipo de acción realizada.
   - 'initialState' → representa el valor inicial del estado que será gestionado por 'useReducer', dicho valor puede ser cualquier tipio de dato .

6. ¿Por qué es útil utilizar useReducer para gestionar el estado en aplicaciones más complejas?
   RTA:

   Es útil y en espacial en aplicaciones complejas, ya que simplifica la gestión de estados estructurados y la lógica de actualización avanzada, ya que, al centralizar la lógica del estado en una función reductora, otorgándole un mayor control, escalabilidad y mantenibilidad al código, teniendo en cuenta también que permite gestionar eficientemente diversas acciones que desencadenan cambios en el estado.

7. ¿Cómo se puede utilizar `React Context` junto con `useReducer` para gestionar el estado global en una aplicación de React?
   RTA:
   Combinar estas dos herramientas es muy eficiente, puesto que permiten gestionar de una manera mas efectiva y eficientes el estado global de un proyecto, especialmente a medida que dichos estados y su lógica se vuelven más complejas.

Para trabajarlos se deben seguir los siguientes pasos:

1. Crear un contexto →Este paso ya se explico anteriormente, pero en esencia nos sirve para transmitir datos entre componentes.
2. Crear un contenedor → globalProvider y se debe envolver la aplicación en este proveedor, para que te pueda utilizar en sus componentes hijos, es decir en lo que estarán dentro de dicho contenedor
3. Consumir el contexto → esto se explicó anteriormente pero básicamente se usa useContext para capturar los datos transmitidos por el contexto

8. ¿Por qué es importante tener un sistema de gestión de estado global en aplicaciones React más grandes?

   RTA:

   Es muy importante porque facilita la comunicación entre los componentes de una aplicación, su mantenimiento y escalabilidad, puesto que, al ser más globales, es más fácil de gestionarlos. Por otro lado, el acceso a datos centralizados, permite la facilidad que tiene cada componente hijo para poder a acceder a la información trasmitida desde el contexto, todo esto es importante, pero tampoco se puede dejar a un lado que evita el abuso excesivo de las props, ya que estas si no existiera el contexto se debería explícitamente pasar por props de componente a componente la información, y si dichos datos sufren alguna alteración corregir todas las props en especial en proyectos complejo o grandes es muy complicado, debido a que al trabajar con props aumente el riesgo de contener código duplicado y es casi que intolerante a cambios, y es ahí donde radica otra importancia, la claridad del código, ya que lo vuelve más amigable y entendibles para las demás personas o colegas que después tomen el código de nosotros.

9. Menciona al menos tres ventajas de utilizar una combinación de `React Context` y `useReducer` en comparación con el manejo de estado local en componentes

   RTA:

   1. Facilita la gestión centralizada del estado dentro de un contexto global, eso ayuda a que no se dependa de estados locales, conllevando así a la simplificación de la actualización y sobre todo al acceso de los datos compartidos.
   2. Evita el exceso de código, en especial el uso de props, puesto que, al tener los datos a nivel globales, ya no se debe pasar explícitamente los datos compartidos de componente a componente, puesto que, estos estas dispuestos para todos los componentes y es decisión de cada cual utilizarlos o no, según sean requeridos
   3. Al tener globalmente definidos los datos y manejos de ellos y su estado, la escalabilidad y mantenibilidad es más fácil, ya que, al tener su estructura centralizada, los cambios o actualizaciones que se deben hacer se ejecutan sobre un pequeño fragmento de código, más no sobre múltiples pedazos del mismo.

10. ¿En qué situaciones podría ser beneficioso migrar de un enfoque de manejo de estado local a un enfoque de estado global utilizando `React Context` y `useReducer`?

    RTA:

    La ventaja mayor es cuando un proyecto empieza a creer, aumentar su complejidad, se desea mejorar su organización, su mantenimiento y empiezan a generarse problemas en la gestión de los estados y comunicación de sus componentes.