AWAKELAB

M4 - DESARROLLO DE INTERFACES INTERACTIVAS CON REACT

DISCUSIÓN Y ANÁLISIS DE CASOS

EQUIPO 2

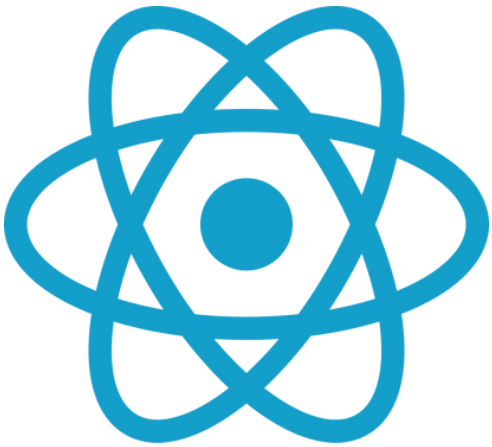
INTEGRANTES:

Felipe Pineda

Marcelo Espinoza

María Fernanda Avello

Karina Hidalgo



Discusión y Análisis de Casos Módulo 4.

1. ReactJS y su Aplicación en el Proyecto del Hospital (1 punto)

Explica qué es ReactJS y justifica su uso en el proyecto del hospital. Argumenta por qué su naturaleza de librería declarativa y su enfoque basado en componentes es ideal para construir una SPA que maneje múltiples secciones (Home, Servicios, Equipo Médico, Contacto) de manera eficiente y dinámica.

¿Qué es React JS? Es una librería de JavaScript que optimiza el desarrollo de una interfaz de usuario web. Sus principales características son:

- **Enfoque basado en componentes:** Permite la asignación de componentes visuales y su comportamiento a objetos y funciones de JavaScript/TypeScript. Esto facilita la reutilización de los componentes visuales.

- **Librería declarativa:** La programación declarativa permite enfocarse en que debe mostrarse y cómo debe hacerlo, sin preocuparse de cómo llegar a ese estado. Este estilo de programación, junto con que React se base en componentes, facilita el trabajo del desarrollador en el layout de la página creada.

En el caso de la página del hospital, la creación de componentes reutilizables, como section, cards, etc. podría ofrecer las siguientes ventajas:

- **Código más legible:** Declarar los componentes dentro de un objeto o función mejora el entendimiento del código.
- **Mayor control del layout:** Los componentes funcionales limitan el acceso a sus parámetros, por lo que al utilizarlo solo hay que indicarle el parámetro requerido y se generará el componente con el layout específico definido para ese componente. Ej: Crear componentes Cards de doctores solo pasando como argumento un objeto doctor, o crear componentes Section que distribuyen los elementos de forma automática según los componentes en su interior.

2. Ventajas de las SPA en un Sistema de Hospital (1 punto)

Analiza por qué utilizar una SPA desarrollada con ReactJS sería beneficioso para el sistema del hospital. Discute la necesidad de una navegación fluida, carga rápida de datos, y la optimización de la experiencia de usuario para acceder a información médica, servicios, doctores, etc., sin recargar la página.

Una SPA (Single Page Application) permite la carga de elementos de forma dinámica a medida que son solicitados por el usuario. Esto evita recargar la página completa, lo que entrega una experiencia de navegación fluida. Además el manejo de estados en React facilita desplegar elementos de carga (por ejemplo progress bar) mostrando un comportamiento más dinámico en los tiempos de espera de adquisición de datos.

3. Manejo del DOM Virtual en la Web del Hospital (1 punto)

Explica cómo el DOM virtual de React puede mejorar el rendimiento de la aplicación web del hospital. Discute cómo la actualización eficiente de los componentes, sin recargar toda la página, puede hacer que el sistema funcione de manera más rápida y eficiente cuando los usuarios navegan entre las distintas secciones (consultas, citas, servicios).

El DOM virtual (VDOM) es un concepto de programación donde una representación ideal o "virtual" de la IU se mantiene en memoria y en sincronía con el DOM "real", mediante una biblioteca como ReactDOM.

Este enfoque hace posible la API declarativa de React: le dices a React en qué estado quieres que esté la IU, y se hará cargo de llevar el DOM a ese estado. Esto abstrae la manipulación de atributos, manejo de eventos y actualización manual del DOM que de otra manera tendrías que usar para construir tu aplicación. Algunas ventajas que permite el VDOM son:

- **Actualizaciones:** Mientras que el DOM tradicional actualiza directamente y puede requerir la redibujo completo de la página, el Virtual DOM de React actualiza sólo los objetos internos hasta que todas las diferencias se han computado, aplicando luego los cambios mínimos necesarios sobre el DOM real.
- **Rendimiento:** El uso del Virtual DOM reduce la carga sobre el navegador, lo que resulta en una mejor velocidad de respuesta y un menor consumo de recursos.
- **Flexibilidad:** React desacopla el estado de la aplicación del DOM, lo que permite manejar más fácilmente la interfaz de usuario y mejorar la experiencia de desarrollo.

Este enfoque no solo optimiza la performance, sino que también simplifica la manera en que los desarrolladores pueden pensar sobre las interacciones y estados en sus aplicaciones, permitiendo centrarse más en la lógica de negocio mientras React se encarga del rendimiento de la interfaz.

Como la información de consultas, citas y/o servicios se carga desde una Base de datos y generalmente no se necesitan al mismo tiempo, se puede cargar dicha información y los componentes que la utilizan de forma parcial, a medida que es requerida. Esto le entregará al usuario una sensación de navegación más fluida.

4. Comparación de ReactJS con Otros Frameworks para el Proyecto (1.5 puntos)

Compara ReactJS con otras tecnologías como Angular o VueJS, y analiza por qué React sería más adecuado para la construcción del sistema del hospital. Considera aspectos como la facilidad de integración con otras herramientas (por ejemplo, bases de datos, API REST para manejo de citas), el manejo del ciclo de vida de los componentes y la modularización.

React se compara frecuentemente con otros frameworks como Angular y Vue. A diferencia de Angular, que es un framework completo, React es más flexible y permite elegir cómo

integrar otras herramientas en la arquitectura de la aplicación. Vue es similar a React, pero ofrece una curva de aprendizaje más sencilla, mientras que React se prefiere en proyectos grandes por su extensibilidad.
Dentro de sus ventajas, podemos decir que es flexible y altamente personalizable, posee una comunidad masiva y excelente soporte y facilita el desarrollo de aplicaciones con requerimientos dinámicos y de alta interacción.

Al elegir una tecnología para construir un sistema hospitalario, es crucial analizar aspectos como la facilidad de integración con APIs y bases de datos, el manejo eficiente de datos dinámicos (como las citas médicas y el historial de pacientes) y la modularización para mantener el código escalable y organizado.

ReactJS sería el más adecuado para desarrollar un proyecto de este tipo porque posee:

a) Facilidad de integración:

- React no impone una estructura rígida, lo que permite una integración sencilla con herramientas externas, como bases de datos (MongoDB, Firebase, SQL) y APIs REST o GraphQL para manejar citas y pacientes.
- La capacidad de trabajar con bibliotecas como **Axios** o **Fetch** para realizar solicitudes HTTP hace que sea fácil consumir APIs REST, esencial para manejar datos dinámicos como horarios de médicos, citas disponibles y reportes médicos.

b) Ciclo de vida de componentes:

- React proporciona un control granular sobre el ciclo de vida de los componentes mediante **hooks** como `useEffect`, lo que facilita la gestión de eventos, actualizaciones y sincronización con datos externos.
- Por ejemplo, al cargar el historial médico de un paciente, `useEffect` puede usarse para realizar una solicitud al API tan pronto como se renderice el componente.

c) Modularización:

- React fomenta el desarrollo basado en **componentes reutilizables**, como componentes de calendario para gestionar citas, formularios para registrar pacientes, y tablas para visualizar historiales médicos.
- Este enfoque modular facilita el mantenimiento y escalabilidad del sistema, especialmente en entornos donde los requisitos evolucionan rápidamente.

d) Ecosistema amplio y flexible:

- Con herramientas como **Redux** o **Context API**, se puede manejar eficientemente el estado global de la aplicación, ideal para manejar datos compartidos como el registro de usuarios, disponibilidad de doctores o notificaciones.

5. Características Clave de ReactJS para el Desarrollo del Hospital (1.5 puntos)

Identifica las características clave de ReactJS que facilitan la implementación del proyecto del hospital:
- Componentización: Cómo dividir la interfaz del hospital en componentes reutilizables.
- Flujo de datos unidireccional: Cómo el control de flujo de información (por ejemplo, desde el backend al frontend) mejora la consistencia del sistema.
- JSX: Cómo facilita la creación de interfaces dinámicas y personalizables, mostrando datos médicos, doctores y citas.

a) Componentización:

Permite crear componentes, los cuales son los elementos visuales que se mostrarán en la aplicación web. En estos componentes se puede definir su disposición, sus características visuales (color, bordes, íconos, etc.) y su comportamiento (animaciones, cambios de tamaño, color, etc.).
Estos componentes se pueden crear mediante clases o funciones, lo que facilita su reutilización para los casos que se requieran. Algunos ejemplos para posibles aplicaciones en la página del hospital serían:

- **Cards:** Se puede crear como un componente funcional, el cual a partir de la información de un doctor (objeto), muestre una card con un tamaño, estilo y disposición definido. Se podría crear una función que itere sobre una lista de doctores y muestre varias cards de forma dinámica.
- **Layouts (Distintos estilos):** Se puede crear como un componente que controle la disposición de los elementos hijo en base a ciertos parámetros de entrada. Podrían hacerle listas para desplegar información, carrousel para fotos, cambiar el fondo, etc.

b) Flujo de datos unidireccional:

En flujo de datos de unidireccional, hace que los datos fluyan desde un componente padre a componentes hijos. Esto tiene algunas limitaciones, pero también tienen las siguientes ventajas.

- Es más fácil de seguir cómo los datos se propagan y cambian a través de la aplicación.
- El código es más fácil de razonar y depurar.
- El mantenimiento de la aplicación es más sencillo.

c) JSX:

La sintaxis de JSX (o TSX), simplifica la definición de los elementos visuales y la interacción con estos. Esta sintaxis crea nuevos elementos (nuevas etiquetas) con sus propios atributos para usarse en el mismo JSX y modificar de forma dinámica atributos, valores y contenido de un elemento html. Además facilita aún más la creación de componentes.

6. Configuración y Ejecución de ReactJS en el Proyecto del Hospital(1 punto)

Explica cómo configurar el entorno de ReactJS para el desarrollo del sistema del hospital. Considera los prerrequisitos, la configuración local o el uso de CDN, y la utilización de herramientas como Webpack, Babel y Redux para manejar la lógica de negocio del hospital (citas, doctores, horarios).

Para comenzar a trabajar con React, se necesitan algunos prerrequisitos:

- Conocimiento de JavaScript ES6.
- Familiaridad con conceptos de componentes y estado.
- Herramientas como Node.js y npm para instalar paquetes y dependencias.

Para proyectos pequeños o de prueba, React puede incluirse directamente desde un CDN (Content Delivery Network). Para proyectos más grandes, se recomienda instalar React en el entorno local utilizando npm o yarn. Esto permite el uso de herramientas como Webpack y Babel para compilar el código.

El comando de instalación con npm es: npm install react react-dom
Esto ha ido cambiando poco a poco, actualmente se recomienda usar vite entre otras opciones para iniciar el desarrollo de un proyecto usando React. El comando es el siguiente: npm create vite@latest nombre-del-proyecto -- --template react
Por último, tenemos la opción de generar un proyecto base de React usando la plantilla por defecto de vite, para eso existe otro comando en el cual podrás controlar paso a paso toda su instalación y las opciones que esta ofrece: npm create vite@latest
Para manejar la lógica de negocio de un hospital utilizando herramientas como **Webpack**, **Babel** y **Redux**, estas desempeñan roles específicos y complementarios en el desarrollo. Podemos mencionar que:

Webpack se encarga de gestionar y empaquetar los recursos del proyecto (JavaScript, CSS, imágenes, etc.) en archivos optimizados para el navegador. Esto es crucial para que el sistema hospitalario sea eficiente y escalable. Permite la agrupación por módulos, organizando los componentes de React para manejar las citas, los horarios y la gestión de doctores de forma modular.

Babel convierte el código moderno (ES6+) a una versión compatible con navegadores más antiguos. En un sistema hospitalario, esto garantiza que todas las funcionalidades trabajen de forma uniforme, incluso en equipos con navegadores obsoletos. Babel asegura que funciones avanzadas como **arrow functions**, **promesas**, y **async/await** funcionen sin problemas. Transforma JSX (usado en React) en JavaScript puro que el navegador puede interpretar.

Redux es ideal para manejar estados compartidos en una aplicación hospitalaria donde múltiples módulos (citas, horarios, doctores) necesitan acceder y actualizar información centralizada. Redux permite una gestión centralizada del estado, centralizando la información de las citas médicas, los doctores y los horarios disponibles. Además utiliza acciones para manejar eventos claves del sistema como agregar una cita, actualizar horarios de un doctor, etc