***Glosario React + JavaScript***

* **React:** Una biblioteca de JavaScript para construir interfaces de usuario, desarrollada por Facebook. React utiliza un enfoque basado en componentes para crear aplicaciones web interactivas.

**Guía básica para crear una aplicación en React:**

*Configura tu entorno de desarrollo:* Asegúrate de tener Node.js y npm instalados en tu sistema. Puedes descargar e instalar Node.js desde [su sitio web oficial](<https://nodejs.org/>).

*Crea un nuevo proyecto de React:* Utiliza Create React App, una herramienta oficial de React para crear aplicaciones con una configuración predeterminada. Para crear un nuevo proyecto, ejecuta el siguiente comando en tu terminal:

**npx create-react-app my-app**

Reemplaza `my-app` con el nombre que desees para tu aplicación.

*Explora la estructura del proyecto:* Una vez que se haya creado tu proyecto, navega hasta el directorio de tu aplicación **(`cd my-app`)** y explora la estructura de archivos generada por Create React App. Esto incluye archivos como **`src/App.js`, `src/index.js`,** **`public/index.html`,** entre otros.

*Desarrolla tu aplicación:* Edita el archivo **`src/App.js`** para comenzar a desarrollar tu aplicación. Este archivo contiene el componente principal de tu aplicación. Puedes modificarlo para agregar tu propia lógica y componentes.

*Ejecuta tu aplicación en modo de desarrollo:* Ejecuta **`npm start`** en tu terminal mientras estás en el directorio de tu aplicación para iniciar el servidor de desarrollo. Esto abrirá automáticamente tu aplicación en tu navegador web predeterminado. A partir de este punto, cualquier cambio que hagas en tu código se reflejará automáticamente en tu navegador.

*Desarrolla y agrega componentes:* Crea nuevos componentes React según sea necesario para tu aplicación. Los componentes son bloques de construcción reutilizables que te permiten dividir tu interfaz de usuario en piezas más pequeñas y manejables.

*Estilo y diseño:* Utiliza CSS, Sass, Styled Components u otra herramienta de estilización para dar estilo a tus componentes y crear una interfaz de usuario atractiva y funcional.

*Gestiona el estado de la aplicación:* Utiliza el estado de los componentes y los hooks de React (como **`useState` y `useEffect`**) para gestionar el estado de tu aplicación y realizar actualizaciones dinámicas en respuesta a eventos o interacciones del usuario.

*Implementa la navegación (opcional):* Si tu aplicación requiere múltiples vistas o páginas, considera usar una biblioteca de enrutamiento como React Router para manejar la navegación entre ellas.

*Prueba y depura tu aplicación:* Realiza pruebas de unidad y de integración para garantizar que tu aplicación funcione correctamente en diferentes escenarios y navegadores. Utiliza herramientas como React Testing Library o Jest para escribir y ejecutar tus pruebas.

*Despliega tu aplicación:* Cuando estés listo para desplegar tu aplicación en producción, ejecuta el comando **`npm run build`** para generar una versión optimizada de tu aplicación que esté lista para ser desplegada en un servidor web. Luego, puedes subir los archivos generados en la carpeta **`build`** a tu servidor de alojamiento web preferido.

* **Componente:** Un elemento de interfaz de usuario reutilizable y modular que encapsula la lógica y la presentación. Los componentes en React pueden ser simples (como un botón) o complejos (como un formulario).
* **Estado (State):** Datos que controlan el comportamiento y la apariencia de un componente. El estado de un componente puede cambiar durante su ciclo de vida en respuesta a acciones del usuario o cambios en los datos.
* **Props (Propiedades):** Valores que se pasan de un componente padre a un componente hijo. Las props permiten que los componentes se configuren dinámicamente y se comuniquen entre sí.
* **JSX:** Una extensión de sintaxis de JavaScript utilizada en React para definir la estructura de la interfaz de usuario. JSX permite escribir código HTML dentro de JavaScript, lo que facilita la creación de componentes.
* **Ciclo de vida del componente:** El conjunto de métodos predefinidos que se llaman automáticamente en diferentes etapas del ciclo de vida de un componente. Estos métodos permiten realizar acciones como inicialización, actualización y eliminación de componentes.
* **Virtual DOM (DOM Virtual):** Una representación virtual de la estructura de un documento HTML en memoria. React utiliza el Virtual DOM para mejorar el rendimiento y la eficiencia al actualizar la interfaz de usuario, minimizando las operaciones en el DOM real.
* **Reconciliación:** El proceso de comparar el Virtual DOM con el DOM real y determinar los cambios necesarios para sincronizarlos. React utiliza un algoritmo de reconciliación eficiente para actualizar solo las partes modificadas de la interfaz de usuario.
* **Eventos:** Acciones del usuario, como hacer clic o escribir en un formulario, que desencadenan cambios en la aplicación. En React, los eventos se manejan utilizando sintaxis similar a JavaScript y se pueden asociar con elementos de la interfaz de usuario.
* **Hooks:** Funciones especiales proporcionadas por React que permiten agregar estado y otras características de React a los componentes funcionales. Los hooks, como useState y useEffect, permiten escribir componentes más simples y reutilizables.
* **Context API**: Es una característica de React que permite pasar datos a través del árbol de componentes sin tener que pasar props manualmente en cada nivel.
* **Vite:** Vite es una herramienta de construcción de proyectos desarrollada por el equipo de Vue.js, pero también es compatible con otras bibliotecas y frameworks como React y Svelte. Vite se enfoca en la velocidad de desarrollo al proporcionar una experiencia de desarrollo rápida y eficiente.

Algunos comandos son:

|  |  |
| --- | --- |
| ***Comando para*** | ***Comando*** |
| Crear un Nuevo proyecto | npm init vite@latest my-project  # O usando yarn  yarn create vite my-project |
| Iniciar el servidor de desarrollo | cd my-project  npm run dev  # O usando yarn  yarn dev |
| Crear una versión optimizada para producción | npm run build  # O usando yarn  yarn build |
| Ejecutar el servidor de desarrollo con HTTPS | npm run dev -- --https  # O usando yarn  yarn dev --https |
| Iniciar el servidor de edesaroollo en un puerto especifico | npm run dev -- --port <puerto>  # O usando yarn  yarn dev --port <puerto> |
| Mostrar la ayuda para los comandos de Vite | npm run dev -- --help  # O usando yarn  yarn dev --help |

* **Node JS:** Node.js es un entorno de ejecución de JavaScript en el lado del servidor construido sobre el motor V8 de Google Chrome. Permite a los desarrolladores ejecutar código JavaScript en el servidor, lo que les brinda la capacidad de crear aplicaciones web dinámicas y escalables.
* **Electron:** Electron es un marco de trabajo de código abierto desarrollado por GitHub que permite a los desarrolladores crear aplicaciones de escritorio multiplataforma utilizando tecnologías web estándar como HTML, CSS y JavaScript