DOXA EDITION

Autor:
GABRIEL HOYOS

2024

Dominando React, una guía introductoria para desarrolladores

Primera Edición

ITQ
WWW.ITQ.EDU.EC
INVESTIGACIÓN

DOMINANDO REACT: GUÍA INTRODUCTORIA PARA DESARROLLADORES

AUTOR: GABRIEL HOYOS

PRIMERA EDICIÓN

AÑO: 2024

TRABAJO EN EDICIÓN:

DIRECCIÓN EDITORIAL: DIEGO JAVIER BASTIDAS LOGROÑO

EDITOR EXTERNO:

ISBN:

DEDICATORIA

Esta obra se encuentra dedicada a mi amada familia, mi amada esposa Katherine mi fortaleza y mi refugio, mis padres Cesar y Lorena quien siempre velan por mí por venir, mi Tía Verónica una segunda madre para mí, mi Primo Ricardo el hermano de mi segunda madre, mis abuelos Cesar y Yolanda, María Elena y Augusto quien siempre están en mi corazón. Todos ellos han formado parte fundamental en mi desarrollo tanto personal como profesional por ello agradezco a Dios tenerlos en mi vida.

AGRADECIMIENTO

Esta obra agradezco al Instituto Superior Tecnológico Quito por ser parte del proceso y apoyo para el desarrollo de esta obra, el agradecimiento especial y de una forma de reconocer y agradecer a cada empresa que me permitió adquirir el conocimiento reflejado en esta obra ya que han contribuido a mi formación profesional a lo largo de los años, también un reconocimiento especial a Dios por hacer posible que esta obra se construya de forma idónea para seguir formando las mentes de futuro.

SOBRE EL AUTOR

Gabriel Hoyos Ingeniero de sistemas graduado en la mención desarrollo de aplicaciones para la gestión en la Universidad Politécnica Salesiana en el año 2018. Su carrera ha estado marcada por una basta trayectoria en proyectos relevantes en el manejo, supervisión, soporte e implementación de ERP's así como desarrollo y diseño de los mismos. Su experiencia y habilidades técnicas le han permitido destacarse en el ámbito de la informática, desarrollo y la tecnología gracias a una sólida comprensión de los sistemas de planificación de recursos empresariales (ERP), con experiencia en la implementación, configuración y personalización de varias plataformas ERP para satisfacer las necesidades específicas de las organizaciones.

He liderado y gestionado proyectos de implementación de ERP desde la fase de planificación hasta la ejecución y el soporte post-implementación, asegurando el cumplimiento de plazos y presupuesto soy experto en trabajar con stakeholders para identificar y documentar requisitos de negocio, lo que me permite adaptar el ERP para mejorar la eficiencia operativa y apoyar los objetivos estratégicos de la empresa utilizado mi conocimiento en ERP´s para analizar y optimizar procesos de negocio, reduciendo costos y aumentando la productividad mediante la automatización y la mejora continua.

Además de proporcionar soporte técnico experto y de capacitación a usuarios finales, ayudándolos a maximizar el uso del sistema ERP y resolver cualquier problema técnico que pueda surgir mediante un profundo conocimiento de las tecnologías y herramientas utilizadas en el desarrollo de interfaces de software, incluyendo lenguajes de programación, bibliotecas, y frameworks de front-end como HTML, CSS, JavaScript, React, Angular, entre otros.

Al ser capaz de descomponer conceptos técnicos complejos en términos más simples que permitan facilitar la comprensión de los estudiantes, utilizando ejemplos claros y relevantes desarrollando planes de estudio que aborden tanto los fundamentos como las tendencias actuales en el desarrollo de interfaces, asegurando que los estudiantes adquieran conocimientos actualizados y aplicables. Lo cual me permite ofrecer orientación y apoyo a los estudiantes, ayudándoles a superar desafíos y a desarrollar sus habilidades de resolución de problemas y pensamiento crítico.

Esto a su vez le ha permitido mantenerse al día con las últimas tendencias y avances en el desarrollo de interfaces de software, incorporando nuevos conocimientos y tecnologías en la enseñanza siendo esta la pauta para comprometerse con el campo del desarrollo y trabajar con otros departamentos y disciplinas que me han permitido integrar el diseño, desarrollo e implementación de interfaces en un contexto más amplio, mostrando cómo se relaciona con otros aspectos dentro del desarrollo de software y la tecnología.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

En este documento vamos a tratar los conceptos básicos del framework React que permitan tener una noción clara de cómo es su estructura de trabajo y la forma y modo en que se emplea en la actualidad. Esta forma de abordar el framework permitirá comprender la documentación inicial y su enorme capacidad para desarrollar interfaces web amigables al usuario, pero sobre todo que sus componentes y cada parte desarrollada con React pueda ser reutilizada dentro del entorno desarrollo especificado para frontend. Es decir, vamos a revisar proceso de desarrollo que van del lado del cliente, como lo es su interfaz mejorada, la amplia gama de funciones que posicionan a React como un marco de trabajo para el desarrollo frontend, cabe mencionar que este es un libro creado con el propósito de comprender la funcionalidad de cada uno de los elementos que se puede crear en React.

En el primer capítulo indagaremos en la historia y la creación de este framework de desarrollo o librería y determinaremos como paso de ser una librería creada en JavaScript a uno de los frameworks más demandados para el diseño de interfaces web, en este proceso de aprendizaje revisaremos también parte de sus avances y como fue cambiando a lo largo de los años hasta ser la principal herramienta de diseño y desarrollo de redes sociales como lo son Instagram y Facebook. En el segundo capítulo veremos la infraestructura de la herramienta y sus principales modos de trabajo por ende revisaremos primero los requerimientos iniciales que se necesitan para poder trabajar y así levantar el ambiente de desarrollo en un equipo ya sea una computadora de escritorio o una laptop.

En el tercer capítulo revisaremos las principales comandos y estructuras que se presentan en el framework de desarrollo una vez instalado y listo para su despliegue y desarrollo de la aplicación a crear, también su modo de despliegue ya que esta herramienta no depende de un servidor web externo, pero sí de otras herramientas que deben tener el equipo de trabajo donde el proyecto se va a generar.

Capítulo 1
Introducción a React una revisión de su historia.

1.1. Creación de la Librería.

React empezó siendo una biblioteca JavaScript para interfaces de usuario de código abierto, especialmente diseñada para crear interfaces de usuario de aplicaciones web. Fue desarrollada inicialmente por Jordan Walke en Facebook en 2011. Desde entonces, se ha convertido en una de las herramientas más populares y utilizadas en el desarrollo frontend, gracias a su eficiencia, flexibilidad y gran ecosistema de herramientas y componentes.

Al paso de los años React paso de ser una librería de JavaScript diseñada para facilitar la creación de interfaces de usuario (UI) interactivas y escalables a un framework de desarrollo frontend. Fue desarrollada por Facebook y es utilizada por muchas grandes empresas, como Instagram y Airbnb, para construir aplicaciones web complejas y de alto rendimiento. Gracias a su arquitectura basada en componentes y enfoque declarativo, React promueve la reutilización de código, simplificando la gestión y el mantenimiento de interfaces de usuario complejas. Además, la amplia comunidad de desarrolladores ofrece un soporte continuo, proporcionando recursos, documentación y herramientas para facilitar el proceso de aprendizaje y desarrollo con React.

React fue creada por un equipo de desarrolladores de Facebook liderado por Jordan Walke. El principal objetivo de Walke era crear una herramienta que simplifique el desarrollo de interfaces de usuario complejas, reduciendo la complejidad y mejorando el rendimiento. Su trabajo se hizo publicó en 2013 y desde entonces, React ha ganado popularidad en la comunidad de desarrollo frontend. (Degni, 2023)

React se crea con el objetivo principal de resolver los desafíos de construir y mantener interfaces de usuario grandes y complejas en aplicaciones web de una manera más eficiente y escalable permitiendo así crear aplicaciones web SPA (Single-Page Application).

1.1.1 SPA modelo o patrón de diseño en la web 2.0

Está claro que la característica más destacada de las SPA es la navegación ágil y fluida. Todo el contenido inicial se carga una única vez, por lo que el tiempo de espera en el paso de una sección a otra es prácticamente inexistente. Estas utilizan menos recursos del servidor, lo que, además de aumentar la velocidad, es beneficioso también para conexiones de Internet lentas. En definitiva, el single-page resuelve uno de los puntos clave de la experiencia de usuario.

Mientras que el flujo de la Múltiple Page Application (MPA) es una aplicación basada en varias páginas por ende indicara los datos del usuario que tiene acceso activo a todas las páginas esto a su vez producirá el siguiente flujo:

1. Se accede a la página;
2. Se activa la solicitud;
3. Se consulta la base de datos;
4. El resultado está formateado (Renderizado);
5. Devuelve la respuesta de visualización al usuario.

Luego navegamos a otra página y se ejecuta el mismo flujo y así sucesivamente, por cada página que se carga. Esto termina siendo malo tanto del lado del servidor como del lado del cliente, precisamente por el flujo repetido de información, dado que cada página va a volver a carga cada elemento nuevamente desde cero esto proporciona una excesiva lentitud en tiempos de respuesta hacia el usuario. Todos los datos de la página serán recargados con cada nueva operación realizada, si el usuario accede a este software usando los datos de internet de su celular, por ejemplo, estará consumiendo mucho más tiempo y datos de su paquete, podrían ser recargados solo si es necesario.

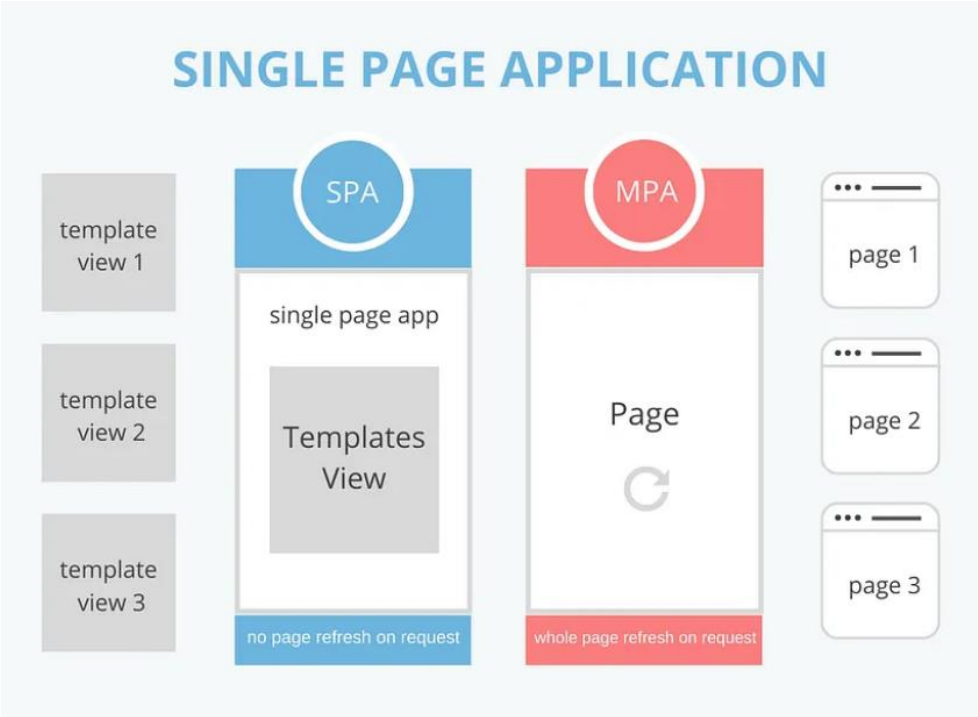
En resumen, dentro de un MPA, cada vista o página de la aplicación es un archivo HTML separado que se carga desde el servidor cuando el usuario navega hasta su equipo esto da como resultado una navegación más lenta y una experiencia de usuario menos fluida, ya que el usuario tiene que esperar a que se vuelva a cargar la página completa cada vez que navega a una página diferente.

Uno de los problemas que tiene el enfoque tradicional (que SPA resuelve fácilmente) es redireccionar a los usuarios a una nueva página cada vez que realizan un cambio en sus datos. Lo que puede hacer que la experiencia sea algo frustrante cuando quieren editar información en diferentes lugares, precisamente porque consume más tiempo y datos. (Ferguson, 2023)

Las aplicaciones basadas en SPA, por otro lado, están totalmente enfocadas en una cosa a la vez. Cuando tenemos una Single Page Application, estamos hablando de aplicaciones cuya funcionalidad se concentra en una sola página, como su nombre lo indica. En pocas palabras, solo se actualizando una parte de la pantalla, sin necesidad de recargar todo el navegador o redirigir al usuario a una nueva página. Por lo tanto, de forma asíncrona, solo se actualiza el contenido principal, manteniendo estática toda la página.

De esta manera, envía y recibe menos datos, lo que resulta en una operación mucho más eficiente. Con una interfaz más rápida y optimización del rendimiento de las aplicaciones. Logrando mejorar la experiencia del usuario (UX), precisamente subiendo su información bajo demanda, con foco en las necesidades del consumidor.

Figura 1:
SPA vs MPA



Fuente: https://medium.com/@teamtechsis/single-page-applications-spa-48b1b845b446

Hay que notar que incluso si nunca has oído hablar de SPA muy probablemente ya hayas utilizado en varias aplicaciones del mundo cotidiano un excelente ejemplo de esto son las plataformas de correos electrónicos que utilizamos hoy en día como Outlook o Gmail. Dentro de ellos es posible abrir un mensaje, borrarlo, responder y solo se recargará la estructura central, manteniéndose el resto de la página estática otros ejemplos de aplicaciones SPA son:

- ✓ Google Maps
- ✓ Trello
- ✓ Facebook
- ✓ Twitter
- ✓ Netflix
- ✓ YouTube

En resumen, la elección entre un SPA y un MPA dependerá de los requisitos y objetivos específicos de su aplicación, hoy en día la mayoría de empresas prefieren una diseño basado en SPA ya que es adecuado para aplicaciones que requieren actualizaciones rápidas y dinámicas, mientras que los MPA son mejores para aplicaciones con una estructura más sencilla y directa como paginas informativas y sin mucha interacción ordenador-usuario por ello es de gran preferencia crear un web o aplicación con este patrón de diseño SPA, todo el código necesario se carga en un solo archivo HTML y la navegación entre diferentes vistas o páginas se maneja dinámicamente en el lado del cliente sin recargar la página completa.

Bien, ya que entendemos cómo funciona SPA y en qué se diferencia de su modelo tradicional. Ahora presentaremos las características más que nos permite aplicar este modelo

1.1.2 Características del Modelo SPA

Almacenamiento en caché de datos (Data Caching)

En mi consideración es una de las características más loables al usar SPA es precisamente el almacenamiento en caché de datos ya que cuando hablamos de este evento, automáticamente tenemos que relacionarlo con la memoria caché del navegador.

La caché del navegador es un espacio donde se almacenan los archivos estáticos de los sitios web visitados. De esa forma, cuando vuelves a la misma página, esta carga es mucho más rápida y permite una mayor agilidad en la navegación.

Cuando intentamos aplicar este concepto dentro del SPA, vemos que la página se puede almacenar en este caché, así logrando acelerar la carga de la aplicación, ya que solo solicitaras el contenido actualizado del servidor.

Performance

En estes apartado lo que se busca es reducir el tráfico de datos entre el servidor y el usuario, es interesante observar cuanto se reduce después de la carga inicial y esto ocurre porque la página ya tendrá la información necesaria para mostrar los datos enviados en paquetes estandarizados y simplificados lo que permite que estos se encuentren ya precargados ayudando a garantizar tiempos de respuesta bajos precisamente por el tamaño reducido de los datos transferidos.

Agilidad de desarrollo

Buscando facilitar el desarrollo de aplicaciones, vemos que SPA cuenta con librerías y frameworks, como actualmente se manejan para diseño todos basados en JavaScript por lo cual elegir una estructura frontend permitirá tener las directrices para continuar con el desarrollo de la aplicación. Hay varias estructuras frontend populares como React, Angular y Vue que son adecuadas para crear SPA. Elige aquel que mejor se adapte a tus necesidades.

Implementar enrutamiento

SPA usa el enrutamiento del lado del cliente para navegar entre diferentes vistas o páginas en la aplicación sin recargar la página completa. Puede usar una biblioteca de enrutamiento como React Router o Angular Router para implementar el enrutamiento en su SPA.

Buscar y mostrar datos

Normalmente SPA se comunica con una API de backend para recuperar y mostrar datos. Use las API proporcionadas por el framework, como obtener React o HttpClient de Angular, para obtener datos y mostrarlos en la página.

Manejar las interacciones del usuario

Implemente lógica para administrar las interacciones de los usuarios, como envíos de formularios y clics de botones, y actualice dinámicamente la página con los datos apropiados.

Probar, depurar e Implementar

Pruebe su SPA a fondo para asegurarse de que funciona como se espera y corrija cualquier error que encuentre. Finalmente, implemente su SPA en un servidor web o plataforma de alojamiento para que los usuarios puedan acceder a él.

Tecnologías que ayudaron a popularizar

Hoy en día, existen varias herramientas, frameworks y soluciones que lo ayudan a usar la aplicación en una sola página. Utilizan HTML5 y AJAX, así como frameworks y bibliotecas como React, Angular y VueJs. El uso de estas herramientas ayudará a compilar la aplicación para que solo represente el fragmento que necesita ser actualizado. Trabajan en la parte de gestión de eventos y muchas veces también consiguen comunicarse con el servidor (en el caso de React a través de librerías, y en frameworks ya viene dentro de la propia solución), que en general, es lo más importante.

React

Desarrollado e introducido por Facebook en 2013 con el fin de mejorar sus estructura y presentación en ese mismo año se dio el cambio a React es la biblioteca de JavaScript más popular, se mantiene por delante de los principales competidores como Angular, proporciona una excelente respuesta para que el usuario agregue comandos utilizando nuevos métodos de representación de sitios web. (Spinola Federico, 2023)

Grandes empresas de todo el mundo utilizan React, por ejemplo:

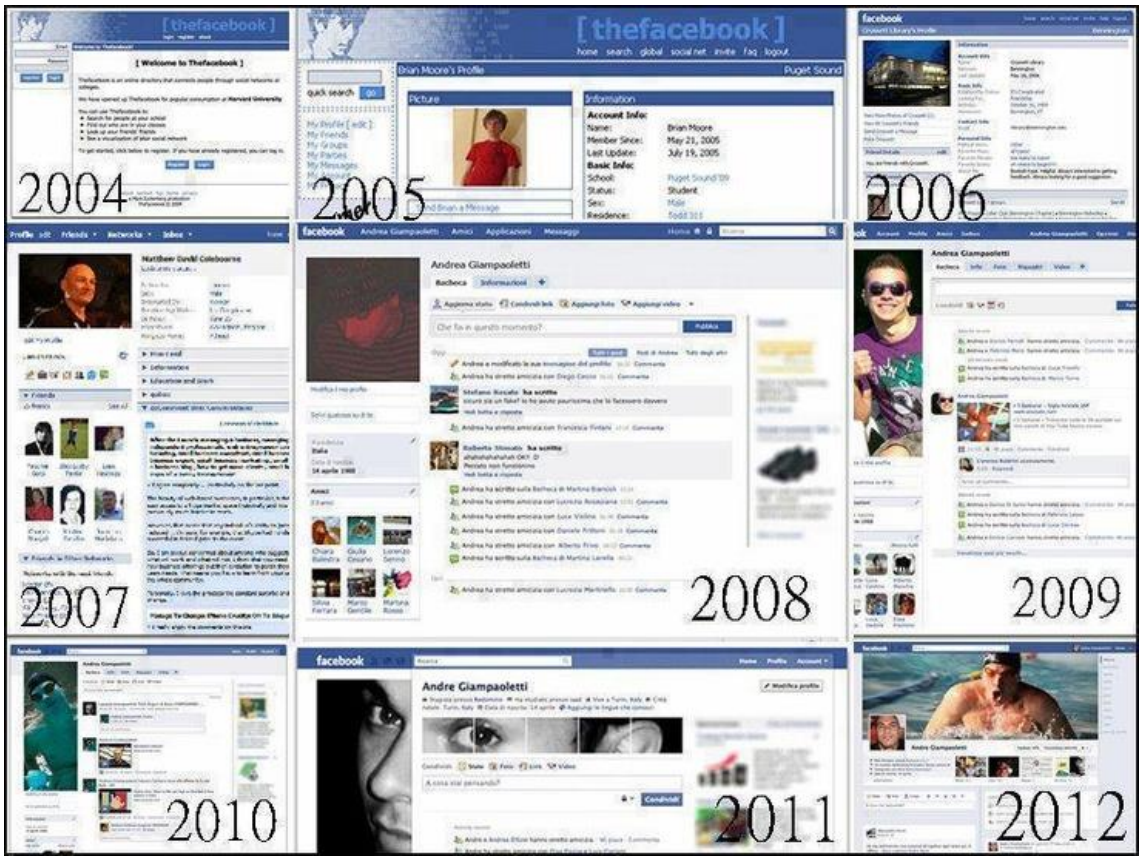
- ✓ Facebook
- ✓ Netflix
- ✓ Airbnb
- ✓ Whatsapp
- ✓ Instagram

Con React, puede crear componentes reutilizables y administrar fácilmente el estado de su aplicación. Además, utiliza un DOM virtual que agiliza las actualizaciones y mejora el rendimiento general. React también se integra bien con otras bibliotecas y herramientas, lo que lo convierte en una opción popular para crear SPA rápidos y atractivos.

Figura 2:
Facebook antes de React

Fuente: https://larepublica.pe/tecnologia/2022/04/24/lo-recuerdas-asi-ha-evolucionado-facebook-durante-toda-su-historia-mark-zuckerberg-fb

Figura 3:
La evolución de FB antes de React



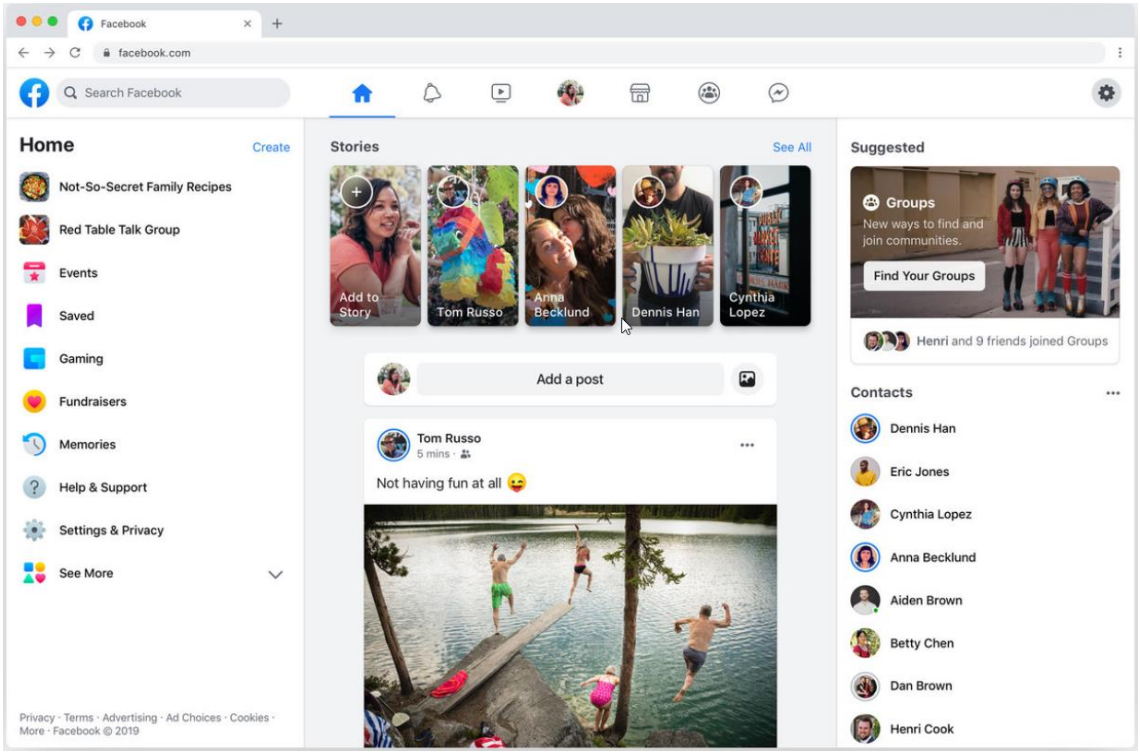
Fuente: https://ar.pinterest.com/pin/269582727666482556/

Aunque muchos no lo recuerdan antes de la creación de componentes las redes sociales era tan planas o más que un correo con la única diferencia que se podían subir fotos en la web en tiempo real es decir que permitían cargar fotos a nuestros perfiles eso con el tiempo fue cambiando a medida del paso de los años el diseño de la red social avanzo a grandes pasos hasta ubicarnos en la creación y al necesidad de tener frameworks de frontend que permitan satisfacer las necesidades sin tener una estructura tan compleja si bien frameworks ya existían antes Facebook decide lanzar React porque era la forma más fácil en acoplar los componentes que ya tenían y mejorar su entorno visual.

Facebook tuvo que esperar 7 largos años para ver cristalizado su ilusión de tener en el flujo de trabajo completo de su red social administrado por la librería que crearon en ese tiempo fue una librería hoy en la actualidad ya es considerado un framework de trabajo Aunque no hay una definición oficial de qué es un framework, la mayoría de la gente considera que uno como tal ya que es una biblioteca que incluye otras bibliotecas para crear una aplicación completa con reglas claras y casi sin configuración. (Durán, 2022)

Por ejemplo, Next.js se podría considerar un framework de React porque incluye React, un sistema de enrutado, un sistema de renderizado del lado del servidor, etc.

Figura 4:
Facebook aplicando componentes con React en el 2020



Fuente: https://www.xataka.com/otros/facebook-redisena-su-aplicacion-pagina-web-al-completo-colores-claros-mayor-importancia-para-grupos

Una gran opción para aquellos que tienen poca experiencia con frameworks y librerías de JavaScript, encargándose de la mayoría de las integraciones necesarias y siendo útil también para proyectos de gran complejidad.

La complejidad de manipular el DOM: El Document Object Model (DOM) es la representación de la estructura de un documento HTML en un objeto programable. Al manipularlo directamente puede ser lento y propenso a errores, especialmente en aplicaciones grandes.

El desafío de mantener la sincronización entre el estado de la aplicación y la interfaz de usuario: A medida que una aplicación crece, mantener una correspondencia exacta entre el estado interno de la aplicación y lo que se muestra en la pantalla se vuelve cada vez más difícil.

1.1.3 React abordo problemas e introdujo una serie de innovaciones

Programación declarativa: En lugar de describir cómo cambiar la interfaz de usuario, React permite a los desarrolladores describir cuál es el estado deseado de la interfaz. React se encarga de aplicar los cambios necesarios de manera eficiente.

Componentes: React divide la interfaz de usuario en componentes reutilizables, lo que facilita la organización del código y la creación de interfaces más modulares.

JSX: es una sintaxis similar a HTML que permite escribir estructuras de UI de forma más intuitiva y concisa dentro de JavaScript.

Virtual DOM: React utiliza un DOM virtual para realizar un proceso comparativo eficiente entre el estado anterior y el nuevo, minimizando las actualizaciones en el DOM real y optimizando el rendimiento. Es decir, una representación del DOM guardada en memoria que no requiere ser visualizada en pantalla por el navegador cada vez que hay un cambio en la interfaz, React compara el virtual DOM con el DOM real para identificar únicamente las partes que deben cambiar en pantalla y solamente modificar esas partes lo que conlleva un ahorro de recursos.

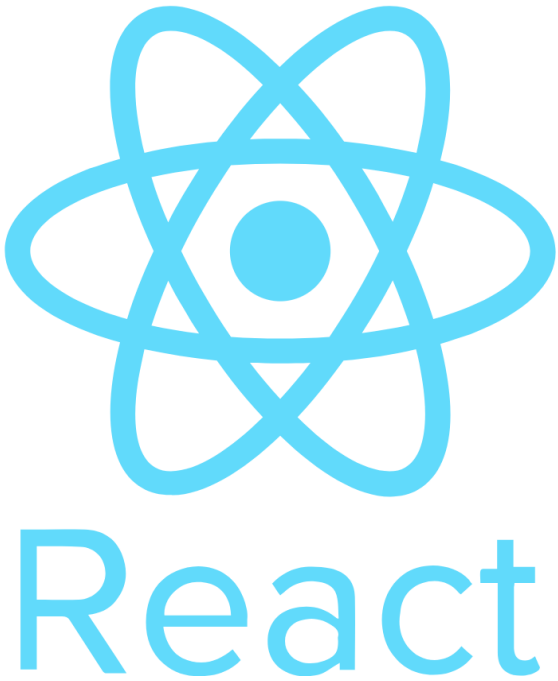
En resumen, los principales motivos detrás de la creación de React son mejorar el rendimiento al utilizar un DOM virtual y un enfoque declarativo, React logra un rendimiento excepcional, especialmente en aplicaciones grandes y complejas permitiendo facilitar el desarrollo de componentes reutilizables y la sintaxis JSX hacen que el desarrollo de interfaces de usuario sea más rápido y sencillo por lo tanto promueve la mantenibilidad a través de la arquitectura de React que fomenta la creación de código más limpio y fácil de mantener.

En conclusión, React se creó para ofrecer una solución más eficiente y escalable a los desafíos de construir interfaces de usuario modernas, y desde entonces se ha convertido en una de las bibliotecas JavaScript más populares y ampliamente utilizadas en la industria, ya que hoy en día cuenta con más herramientas a su cargo por lo cual puede ser también ya considerado como un framework de trabajo frontend.

1.2 Logo Atómico

La asociación entre React y la representación visual de un átomo es una interpretación que ha surgido de forma orgánica en la comunidad de desarrolladores y no está respaldada oficialmente por el equipo de React.

Figura 5:
Logo de React



Fuente: https://es.react.dev/learn

A continuación, detallamos las razones por las cuales fue elegido ese logo:

1.2.1 Metáfora de la composición

La idea de que los componentes de React se combinan para formar interfaces de usuario más grandes, de forma similar a como los átomos se combinan para formar moléculas, ha sido utilizada como una metáfora pedagógica para explicar los conceptos fundamentales de la biblioteca.

1.2.2 Diseños personalizados

La comunidad de desarrollo ha creado logotipos y diseños personalizados para React, algunos de los cuales han incorporado elementos atómicos como una forma de expresar su propia interpretación de la biblioteca

1.2.3 Abstracción y versatilidad

Es una herramienta de desarrollo altamente versátil, utilizada para construir una amplia variedad de aplicaciones. Un logotipo demasiado literal, como uno que representa un átomo, podría limitar la percepción de React y no reflejar su capacidad para adaptarse a diferentes contextos.

1.2.4 Evolución constante

El ecosistema de desarrollo web evoluciona rápidamente, y los logotipos de las tecnologías también deben adaptarse a estos cambios. Un logotipo demasiado específico podría dificultar futuras rediseños o actualizaciones de la marca. Su enfoque se encuentra en la funcionalidad por ello el equipo de React se centra en el desarrollo y mejora continua de la biblioteca, y el diseño del logotipo es una consideración secundaria.

En conclusión, la asociación entre React y el átomo es una interpretación visual aunque carece de un fundamento oficial. La elección de un logotipo para una tecnología como React implicó una serie de consideraciones estratégicas y de diseño que van más allá, la idea era una simple representación visual del proceso que conlleva tener varios componentes.

1.3 La Evolución en React

React, desde su concepción en los laboratorios de Facebook, ha experimentado una transformación significativa, consolidándose como una de las bibliotecas JavaScript más influyentes para el desarrollo de interfaces de usuario. Su evolución ha sido marcada por una serie de hitos técnicos y conceptuales que han moldeado el panorama del desarrollo frontend.

La evolución de React, una biblioteca JavaScript creada que por sus avances bien ya podría ser considera un framework para construir interfaces de usuario, ha sido significativa desde su lanzamiento. A continuación, detallo para mi forma de ver una cronología de los hitos más importantes en la historia de React:

- ✓ 2011:

  Origen de React: React fue creado inicialmente por Jordan Walke, un ingeniero de software en Facebook, como una herramienta interna para simplificar el desarrollo de la interfaz de usuario.

  Gestación y Prototipado: Concebido internamente en Facebook como una solución a los desafíos de construir y mantener interfaces de usuario complejas, React emerge como un prototipo que promete una nueva forma de abordar el desarrollo frontend.

  - Hito: Aunque no se lanzó públicamente en 2011, este año marca la creación de la primera versión de React, llamada "FaxJS".

- ✓ 2013:

  Lanzamiento de React al público: Se lanzó oficialmente al público en la conferencia JSConf US en mayo de 2013.

  Nacimiento como Código Abierto: Facebook libera React al público, democratizando el acceso a esta tecnología y fomentando una comunidad de desarrolladores en torno a ella. Este hito marca el inicio de una adopción masiva y el establecimiento de un ecosistema rico y diverso.

  - Hito: React introdujo el concepto revolucionario del Virtual DOM, permitiendo una actualización eficiente de la interfaz de usuario sin necesidad de redibujar toda la página web, Es decir un renderizado en tiempo real si yo realizo un cambio debe reflejarse inmediatamente en el navegador.

- ✓ 2015:

  React Native: Facebook lanzó React Native, permitiendo a los desarrolladores usar React para crear aplicaciones móviles nativas para iOS y Android.

  Expansión a Móvil con React Native: La introducción de React Native extiende el alcance de React al desarrollo de aplicaciones móviles nativas, permitiendo a los desarrolladores compartir una base de código entre plataformas web y móviles.

  - Hito: Este fue un paso clave para React, extendiendo su aplicación más allá del desarrollo web y consolidándolo como una herramienta versátil para múltiples plataformas.

- ✓ 2016:

  React Fiber: fue una reescritura interna de React que mejoró significativamente su rendimiento.

  - Hito: Aunque Fiber se completó en 2017, su desarrollo comenzó en 2016. Fiber introdujo un sistema de actualización más flexible y eficiente, permitiendo interacciones de usuario más suaves en aplicaciones complejas.

- ✓ 2017:

  Lanzamiento de React 16.0 : Es la versión de React que incorporó React Fiber y trajo mejoras como la capacidad de renderizar múltiples elementos sin un contenedor adicional (Fragmentos) y un mejor manejo de errores con los "Error Boundaries".

- Hito: Esta versión marcó una evolución significativa en la robustez y flexibilidad de React.

✓ 2018:

React Hooks: En 2018, React introdujo los Hooks en la versión 16.8.

Revolución con Hooks: La presentación de los hooks representa un cambio paradigmático en la forma de gestionar el estado y los efectos secundarios en componentes funcionales de React, simplificando la lógica del componente y mejorando la legibilidad del código.

- Hito: Los Hooks permitieron a los desarrolladores usar el estado y otras características de React sin tener que escribir una clase, simplificando aún más la sintaxis y mejorando la reutilización de código.

✓ 2020:

React 17: Se lanzó con un enfoque en mejorar la compatibilidad y la facilidad de actualización.

- Hito: Aunque no introdujo nuevas características importantes, React 17 preparó el camino para futuras actualizaciones sin rupturas, facilitando la migración gradual en grandes aplicaciones.

✓ 2022:

React 18 y Renderizando Concurrente: Se lanzó con mejoras significativas en la capacidad de renderizado concurrente.

- Hito: Este lanzamiento introdujo la API de useTransition, y otros mecanismos para manejar la carga de trabajo de renderizado de manera más eficiente, mejorando la experiencia del usuario en aplicaciones complejas.

La evolución de React ha sido un viaje apasionante, marcado por innovaciones técnicas y una creciente adopción por parte de la comunidad de desarrolladores. Su enfoque en la simplicidad, la eficiencia y su modularidad lo ha convertido en una herramienta indispensable para construir aplicaciones web modernas y escalables.

1.4 React ¿Framework o Librería?

En el ámbito del desarrollo de software, tanto los frameworks como las librerías son herramientas esenciales que facilitan la construcción de aplicaciones. Sin embargo, presentan diferencias fundamentales en cuanto a su alcance, estructura y control que el desarrollador ejerce sobre el proyecto.

1.4.1 Framework

Un framework proporciona una estructura completa y predefinida para desarrollar aplicaciones. Impone una arquitectura específica, define convenciones de nombrado y ofrece un conjunto de herramientas y componentes reutilizables que abarcan todo el ciclo de desarrollo. El desarrollador se adapta a esta estructura, siguiendo las pautas establecidas por el framework. Esto proporciona una base sólida y consistente, pero limita la flexibilidad en ciertos aspectos. Es decir, proporciona una estructura completa para desarrollar aplicaciones. Define una arquitectura, patrones de diseño y un conjunto de herramientas que guían todo el proceso de desarrollo. Es como un andamio que te indica cómo construir un edificio, pero te da mucha libertad para diseñar los detalles internos.

1.4.2 Librería

Una librería, por su parte, es una colección de funciones o componentes que resuelven problemas específicos. No impone una estructura rígida al proyecto, sino que proporciona bloques de construcción que el desarrollador puede utilizar de forma selectiva y combinarlos según sus necesidades. Esto otorga al desarrollador un mayor control sobre la arquitectura de la aplicación, pero también requiere una mayor toma de decisiones. Es decir, son soluciones dentro del desarrollo de software que pueden ser reutilizables y que resuelven problemas específicos. Es como una caja de herramientas que puedes utilizar para realizar tareas concretas dentro de tu proyecto. Te ofrece bloques de construcción, pero no impone una estructura rígida.

React es clasificada como una librería ya se enmarca claramente dentro de la categoría de librería debido a las siguientes razones:

Enfoque en la capa de vista: React se especializa en la construcción de interfaces de usuario, proporcionando herramientas para crear componentes reutilizables y gestionar el estado de la aplicación. Sin embargo, no abarca otros aspectos fundamentales de una aplicación web, como el enrutamiento, la gestión de datos o la interacción con el servidor.

Flexibilidad: React ofrece una gran flexibilidad al desarrollador, permitiéndole elegir otras librerías y herramientas para complementar sus funcionalidades. Esto contrasta con los frameworks, que suelen imponer un conjunto de tecnologías y patrones de diseño específicos.

Modelo de componentes: React promueve un modelo de programación basado en componentes, donde la interfaz de usuario se descompone en piezas más pequeñas y reutilizables. Este enfoque facilita la creación de interfaces complejas y mantenibles.

Sin Imposición: React no impone una opinión sobre cómo debe estructurarse una aplicación, lo que permite al desarrollador adaptar la librería a diferentes estilos arquitectónicos y flujos de trabajo.

En conclusión, React se distingue como una librería debido a su enfoque modular, su flexibilidad y su capacidad para ser integrado en diferentes tipos de aplicaciones. Si bien proporciona una base sólida para construir interfaces de usuario, deja al desarrollador un amplio margen de maniobra para diseñar la arquitectura general de la aplicación.

En resumen, las principales diferencias entre frameworks y librerías se pueden resumir en los siguientes puntos:

Tabla 1:
Framework vs Librería una comparativa técnica.

Característica	Framework	Librería
Estructura	Impone una estructura rígida de desarrollo	Ofrece componentes reutilizables
Control	Menos control sobre la arquitectura	Mayor control sobre la implementación
Alcance	Resuelve problemas más amplios	Resuelve problemas específicos
Ejemplo	Angular, Ruby on Rails	jQuery, React

Esta diferenciación es fundamental para elegir la herramienta adecuada para cada proyecto, teniendo en cuenta las necesidades específicas y las preferencias del equipo de desarrollo.

Oficialmente, React se autodenomina como biblioteca en su sitio web oficial. Esto es porque para poder crear una aplicación completa, necesitas usar otras bibliotecas, y se considera que React se enfoca específicamente en la construcción de interfaces de usuario. Por ejemplo, React no ofrece un sistema de enrutado de aplicaciones oficial. Por ello, hay que usar una biblioteca como React Router o usar un framework como Next.js que ya incluye un sistema de enrutado. (Durán, 2022)

Otra diferencia es que React no te fuerza en qué empaquetador de aplicaciones usar. En cambio, Angular en su propio tutorial ya te indica que debes usar @angular/cli para crear una aplicación, en cambio React siempre te deja la libertad de elegir qué empaquetador usar y ofrece diferentes opciones.

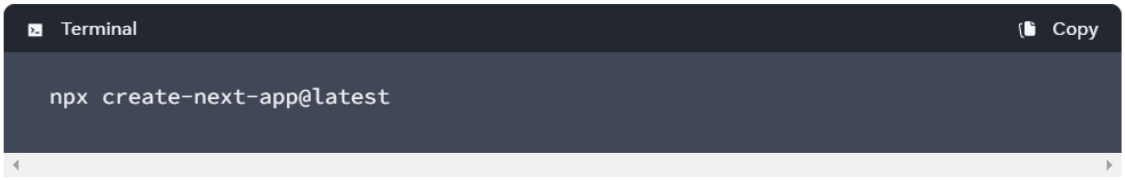
Aun así, existe gente que considera a React como un framework. Aunque no hay una definición oficial de qué es un framework, la mayoría de la gente considera que un framework es una biblioteca que incluye otras bibliotecas para crear una aplicación completa con reglas claras y casi sin configuración. (Durán, 2022) Por ejemplo, Next.js se podría considerar un framework de React porque incluye React, un sistema de enrutado, un sistema de renderizado del lado del servidor.

1.5 Frameworks React a nivel de producción

Estos frameworks brindan todas las características que necesitas para implementar y escalar tu aplicación en producción y están trabajando para respaldar la visión de una arquitectura fullstack. Todos los frameworks que la página oficial de React recomiendan son de código abierto, con comunidades activas que brindan soporte, y pueden ser desplegados en tu propio servidor local o en un proveedor de alojamiento (host), además los mismos desarrolladores de la herramienta solicitan ayuda en general si se ha establecido un nuevo enfoque en frameworks para que se los comuniquen y puedan agregarlo.

- ✓ **Next.js' (Pages Router)**: Es un framework completo de React. Es versátil y te permite crear aplicaciones React de cualquier tamaño, desde un blog mayormente estático hasta una aplicación dinámica compleja. Para crear un nuevo proyecto de Next.js, ejecuta en tu terminal: npx create-next-app@latest (Facebook, 2013)

Figura 6:
Comando de Creación Next JS en una terminal.

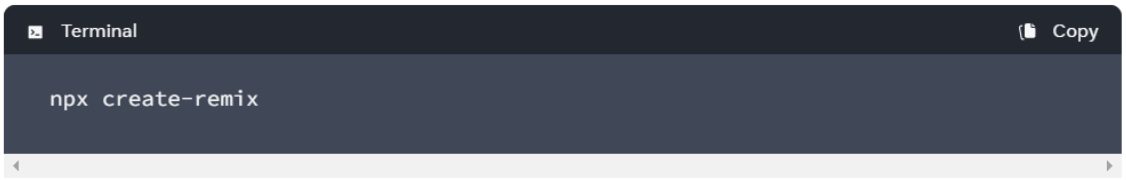


Fuente: https://es.react.dev/learn/start-a-new-react-project

El mantenimiento de Next.js está a cargo de Vercel. Puedes implementar una aplicación Next.js en cualquier alojamiento de Node.js, serverless, o en tu propio servidor. Next.js también admite una exportación estática que no requiere un servidor.

- ✓ **Remix**: Es un framework de React muy completo con enrutamiento anidado. Te permite dividir tu aplicación en partes anidadas que pueden cargar datos en paralelo y actualizarse en respuesta a las acciones del usuario. Para crear un nuevo proyecto Remix, ejecuta: npx create-remix (Facebook, 2013)

Figura 7:
Comando de Creación Remix en una terminal.

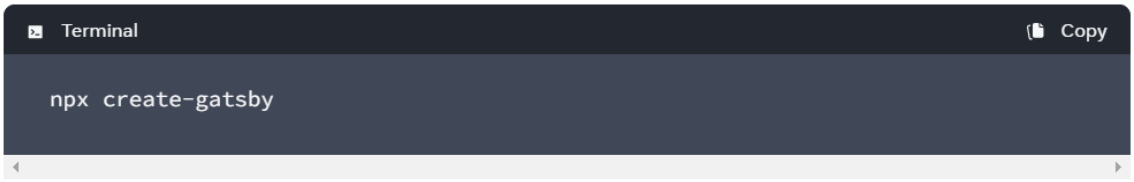


Fuente: https://es.react.dev/learn/start-a-new-react-project

Remix es mantenido por Shopify. Cuando creas un proyecto Remix, debes elegir su destino de implementación. Puedes implementar una aplicación Remix en cualquier ambiente Node.js, alojamiento sin servidor usando o escribiendo un adaptador.

- ✓ **Gatsby**: Es un framework de React para sitios web rápidos respaldados por un CMS (Un sistema de gestión de contenidos o gestor de contenidos). Su ecosistema es rico en complementos y su capa de datos GraphQL simplifica la integración de contenido, API y servicios en un sitio web. Para crear un nuevo proyecto de Gatsby, ejecuta: npx create-gatsby

Figura 8 :
Comando de Creación Gatsby en una terminal. '

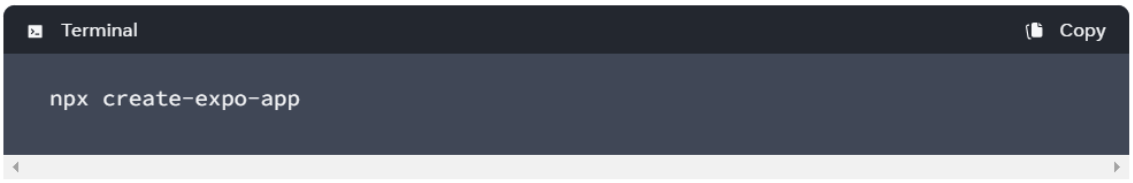


Fuente: https://es.react.dev/learn/start-a-new-react-project

Gatsby es mantenido por Netlify. Puedes implementar un sitio estático de Gatsby en cualquier alojamiento estático. Si optas por usar funciones solo del servidor, asegúrate de que tu proveedor de alojamiento las admita para Gatsby.

- ✓ **Expo**: Es un framework de React que te permite crear aplicaciones web, iOS y Android universales con interfaces de usuario verdaderamente nativas. Proporciona un SDK para React Native que facilita el uso de las partes nativas. Para crear un nuevo proyecto Expo, ejecuta: npx create-expo-app

Figura 9:
Comando de Creación Expo en una terminal.



Fuente: https://es.react.dev/learn/start-a-new-react-project

Expo es mantenida por Expo (empresa). La creación de aplicaciones con Expo es gratuita y puede enviarlas a las tiendas de aplicaciones de Google y Apple sin restricciones. Expo también ofrece servicios en la nube de pago opcionales.

Frameworks React de última generación

A medida que exploramos cómo continuar manejando React, nos damos cuenta de que la integración de React se vuelve más estrecha con los frameworks (específicamente, con tecnologías de enrutamiento, agrupación y servidor) lo que conlleva una mayor ayuda hacia a los usuarios de React para la creación de mejores aplicaciones. Actualmente React tiene una herramienta más que es la prueba de funciones de React de última generación independientes del framework lo es Vite y en el cual profundizaremos en posteriores capítulos a su vez esta herramienta nos facilitara el correcto uso de los React Server Components.

Estas funcionalidades están cada día más cerca de ver la luz y por ello era la pregunta si esta librería dejo de serlo como tal y paso a una nueva etapa del desarrollo pude que en futuro estén listas para producción, ya que los desarrolladores de Facebook, así como de Next han mantenido conversaciones con otros desarrolladores de paquetes y frameworks para integrarlas en uno solo. Esperamos que en el trascurso de los años todos los frameworks que aparecen sean totalmente compatibles con estas funciones.

Resumen del Capítulo 1

En el primer capítulo de este libro hemos visto una breve reseña de cada hito importante en las historia de la librería/framework React , lo cual nos permitió revisar la descripción del contenido sobre la creación de modelos de páginas web como lo son SPA y MPA estas estructuras son parte de los temas principales que trata React de abordar y mejorarlas en el capítulo también hicimos una breve reseña histórica que permitió comprender como y el por qué se creó esta librería ya que la empresa Facebook decidió optimizar el proceso de diseño y creación de su red social y presentar al mundo una alternativa que permita diseñar de forma eficiente , practica y sencilla para el diseño de interfaces web esto gracias al grupo visionario de desarrollo liderado por Jordán Walke el cual busco proporcionar una visión general sobre el diseño y sus distintos enfoques y metodologías para desarrollar la red social, y no solo esta sino muchas otras como lo hemos visto permitiendo así tener una clara idea para la mejora continua de resultados en las aplicaciones que manejamos en la actualidad.

En si podemos darnos cuenta de el correcto uso de los componentes del servidor y sus funcionalidades permiten a React, ser una de las librerías/frameworks mas adaptativos lo que requiere compromiso y un trabajo de implementación no trivial. Por el momento, es una librería más completa para el diseño web de interfaces de usuario (UI). A su vez hay que mencionar que el equipo de React está trabajando con los desarrolladores de paquetes para que estas características sean más fáciles de implementar en la próxima generación de frameworks y así esta librería del salto de calidad tan esperado, aun que como bien indique en apartados anteriores del capítulo varias personas ya lo tratan como un framework de desarrollo frontend. (Azaustre, 2023)

Facebook, así como de Next han mantenido conversaciones con otros desarrolladores de paquetes y frameworks para integrarlas. Esperamos que en transcurso del tiempo aparezca una estructura mucho más definida y que sean totalmente compatibles con estas funciones, a partir de la evolución de React hemos podido ver cada uno de las mejoras y avances que se han desempeñado una mejora continua en el diseño de interfaces lo cual permite ver en tiempo real los cambios a esto se le denomina el renderizado de una aplicación.

Por último, pero no menos importante hemos revisado la clasificación actual de frameworks que trabajan con React y permiten su creación esto es el pie de inicio en nuestro camino con React el saber implementar la herramienta que se va a usar para diseñar la web y que framework es el que nos beneficia para su mantenimiento a lo largo del tiempo, es así que manteniendo esta tendencia podremos crear aplicaciones web llamativas y con una librería que está en auge en el ámbito del desarrollo. (Chan, 2016)

Capítulo 2
Instalación y Administración de React.

2.1 Prerrequisitos para el Entorno de Desarrollo de React.

Dentro del proceso de instalación de la librería se consideran que el equipo de trabajo donde vamos a iniciar con nuestros proyectos debe estar configurado y debe tener instalado los siguientes prerrequisitos para establecer un entorno de desarrollo óptimo para React, es fundamental contar con los siguientes componentes:

2.1.1 Gestor de Paquetes de Nodo (npm o yarn):

npm: Incluido por defecto con Node.js, es el gestor de paquetes estándar para el ecosistema JavaScript. Se utiliza para instalar, actualizar y gestionar las dependencias de tu proyecto React.

yarn: Una alternativa a npm, conocido por su velocidad y confiabilidad en la gestión de dependencias (basadas para Sistemas Operativos Unix Ej.: Ubuntu).

2.1.2 Node.js:

Un entorno de ejecución de JavaScript que permite ejecutar código JavaScript fuera del navegador. Es esencial para la instalación y ejecución de herramientas de desarrollo de React.

2.1.3 Editor de Código:

Un editor de código potente es crucial para una experiencia de desarrollo eficiente. Algunas opciones populares incluyen:

- ✓ **Visual Studio Code**: Es una excelente opción, muy popular entre los desarrolladores ampliamente utilizado en la comunidad React debido a su gran cantidad de extensiones y personalización, fue creado por Microsoft.
- ✓ **Sublime Text**: Un editor de texto ligero y altamente personalizable.
- ✓ **Atom**: Desarrollado por GitHub, ofrece una interfaz intuitiva y una gran comunidad de extensiones.

2.1.4 Navegador Web Moderno:

Para visualizar y depurar tus aplicaciones React, necesitarás un navegador web que soporte las últimas especificaciones de JavaScript. Chrome, Firefox, Edge, Safari o Brave son opciones comunes.

2.2 Proceso de Instalación

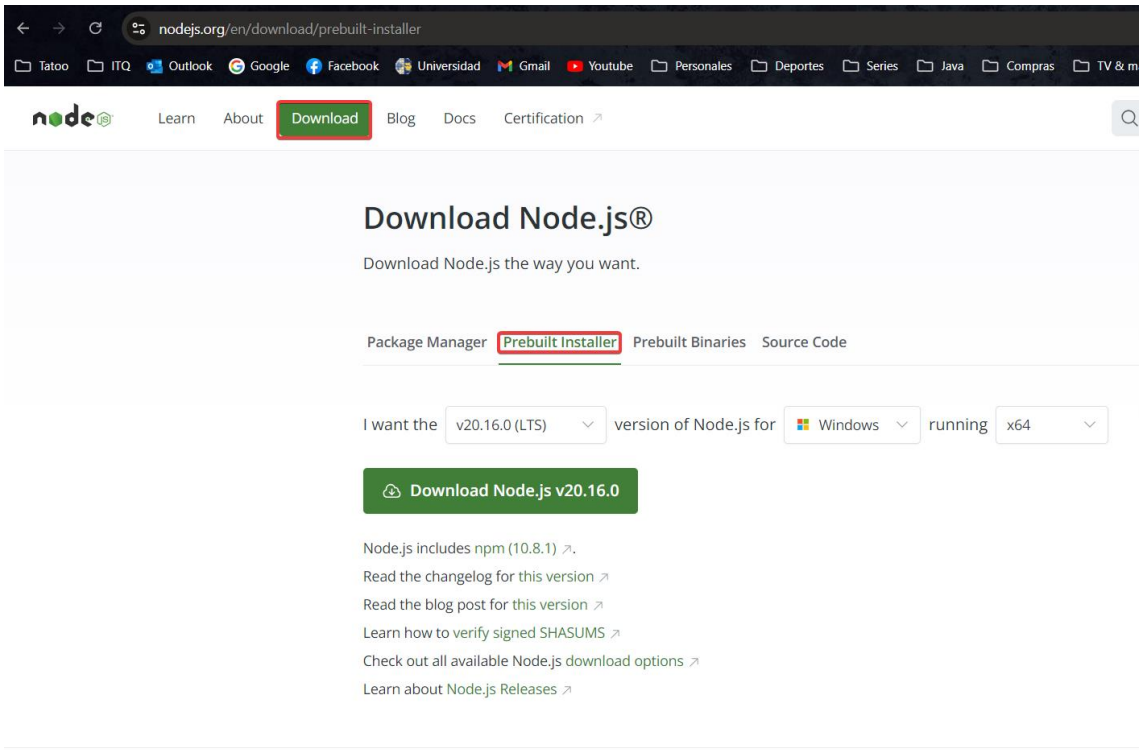
Aquí revisaremos la instalación de Node.js paso a paso desde el proceso de descarga de la página oficial de NODE .JS (https://nodejs.org/en/download/prebuilt-installer) hasta su despliegue en el equipo a instalar por ello hay que seguir las instrucciones que se encuentra detalladas a continuación.

2.2.1 Node.js y npm:

Necesitamos dirigirnos a la página oficial de NodeJs (desde https://nodejs.org/en/download/prebuilt-installer) en la misma encontraran el apartado de descargas donde encontraremos la sección de prebulit, Es decir un instalador de Node.js para en este nuestro caso el sistema operativo Windows de 64 bits.

Una vez descargado procedemos a instalar el MSI los cuales son paquetes de instalación de la aplicación y se definen como instaladores de Microsoft, es decir todos aquellos paquetes de software que contienen la información necesaria para automatizar su instalación, minimizando la intervención manual del usuario, ya que toda la información iría contenida en el propio archivo o fichero ".msi". Cabe recalcar que el empaquetado de npm se instala automáticamente junto con Node.js.

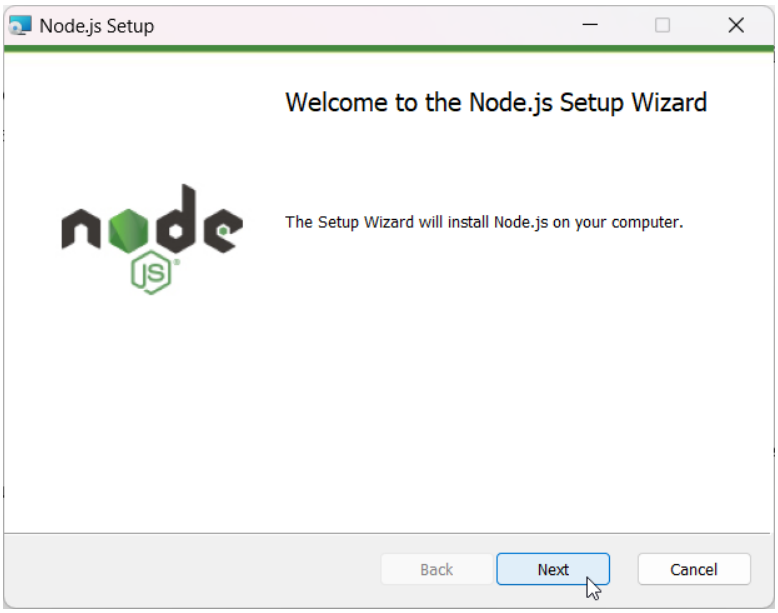
Figura 10:
NodeJs para descarga versión LTS



Fuente: https://nodejs.org/en/download/prebuilt-installer

Luego nos posicionaremos sobre el instalador damos clic derecho sobre el mismo y seleccionamos la opción ejecutar/instalar con permisos de administrador por lo que debe salir un menú de instalación como imagen de la figura 11.

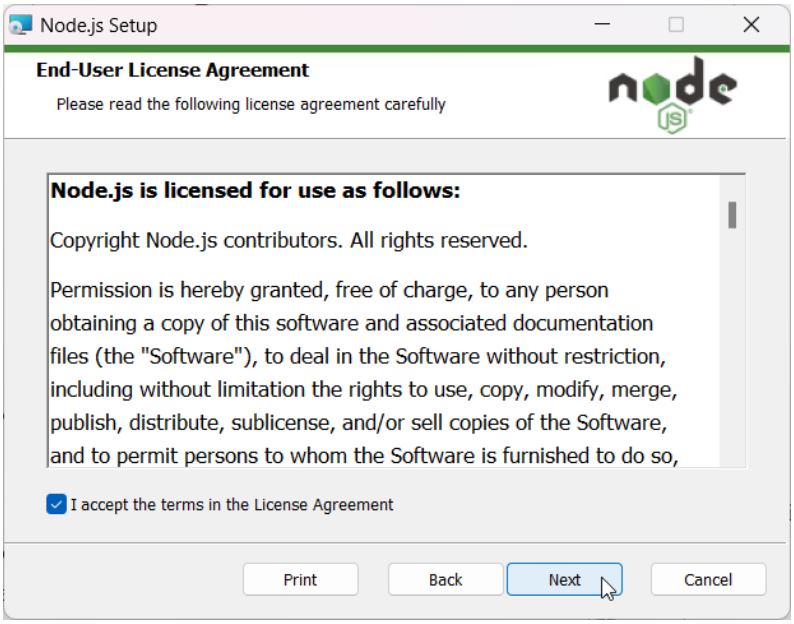
Figura 11:
Instalacion de NodeJs en Microsft Windows 11 Parte 1



Elaborado por: Autor

Luego nos posicionaremos en la siguiente ventana de instalación y le damos clic en el check que acepta todos los términos de la licencia de instalación y damos clic en siguiente como en la imagen de la figura 12.

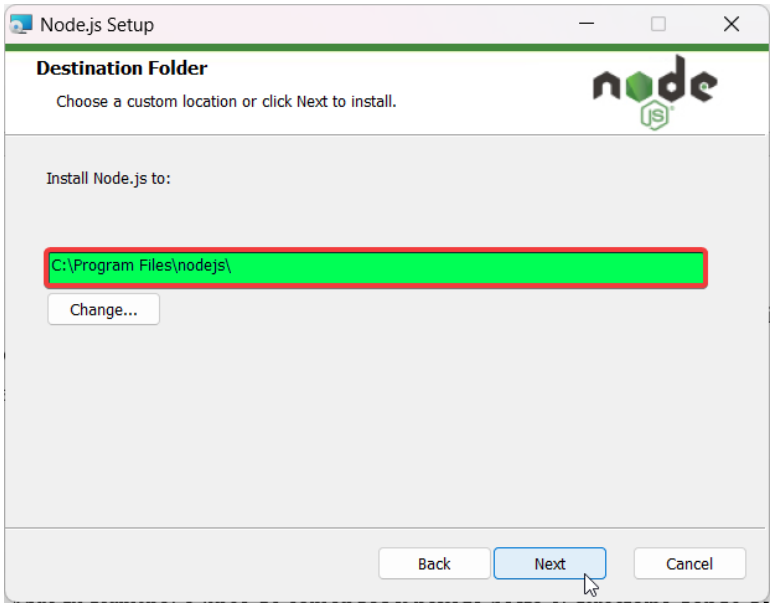
Figura 12:
Instalacion de NodeJs en Microsft Windows 11 Parte 2



Elaborado por : Autor

Luego nos posicionaremos en la siguiente ventana de instalación donde podemos divisar la ruta de instalación de NodeJs esto lo puedes cambiar si lo cree necesario ya que por defecto lo ubica en el disco local C y damos clic en siguiente como en la imagen de la figura 13.

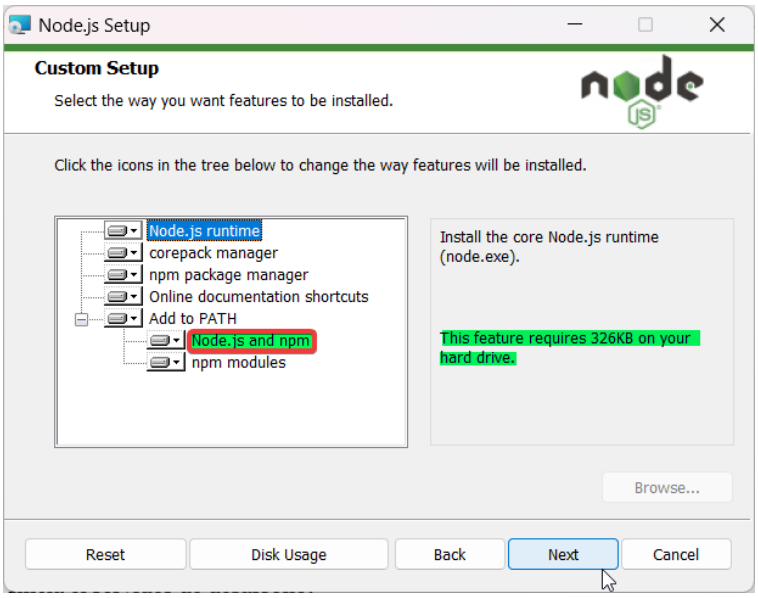
Figura 13:
Instalacion de NodeJs en Microsft Windows 11 Parte 3



Elaborado por: Autor

Posterior a esto el instalador de NodeJs nos indica cuanto espacio se va a necesitar en la instalación del disco local y que paquetes de instalación se están ejecutando, damos clic en siguiente como en la imagen de la figura 14

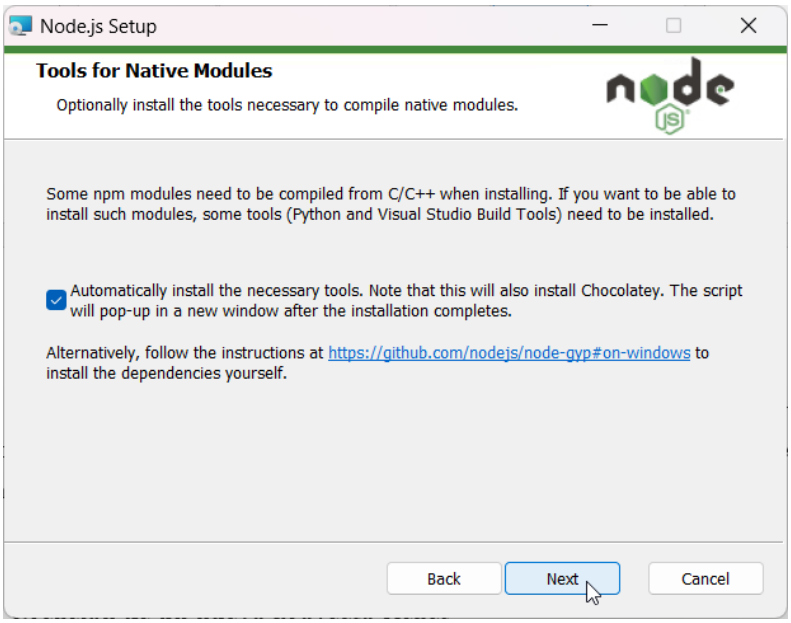
Figura 14:
Instalacion de NodeJs en Microsft Windows 11 Parte 4



Elborado por: Autor

Luego nos posicionaremos en la siguiente ventana de instalación y le damos clic en el check que acepta la instalación de herramientas complementarias como lo es chocolate y damos clic en siguiente como en la imagen de la figura 15.

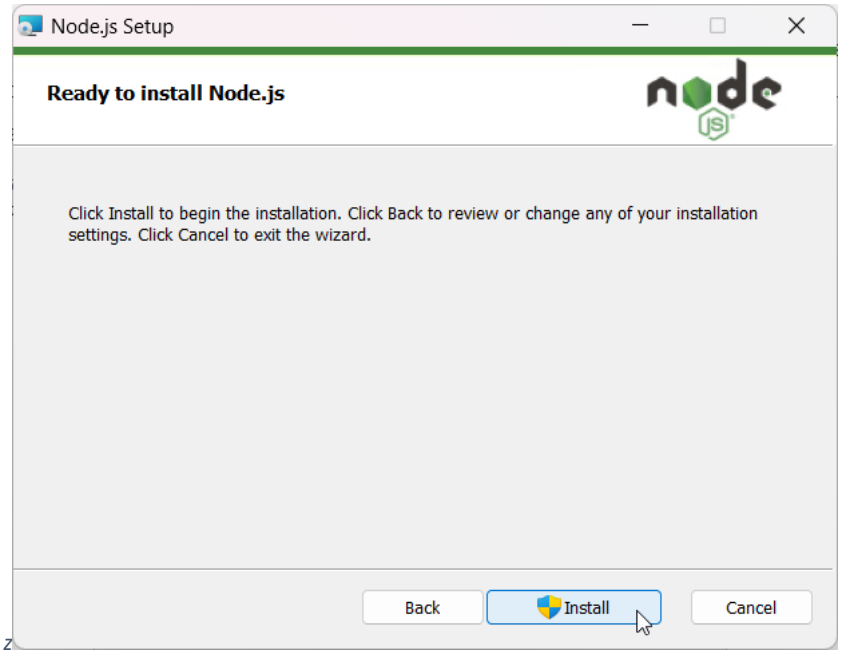
Figura 15:
Instalacion de NodeJs en Microsft Windows 11 Parte 5



Elaborado por: Autor

Luego nos posicionaremos en la siguiente ventana final de la configuración para la instalación y damos clic en el botón de install como en la imagen de la figura 16.

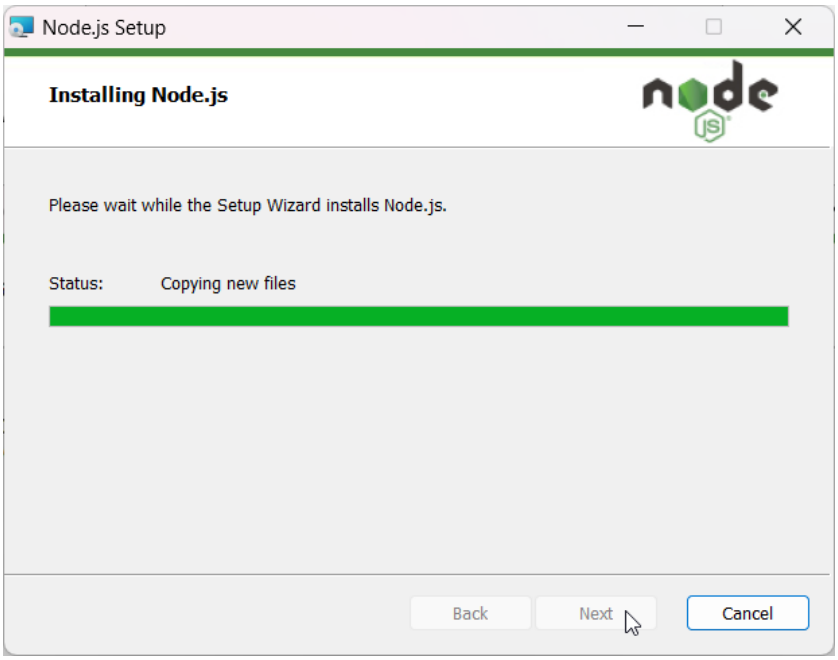
Figura 16:
Instalacion de NodeJs en Microsft Windows 11 Parte 6



Elaborado por: Autor

Luego nos posicionaremos en la siguiente ventana final de la instalación de NodeJs y esperamos que la barra de avance se complete en este caso si no deseáramos instalarlo podríamos dar clic en el botón de cancel como en la imagen de la figura 17.

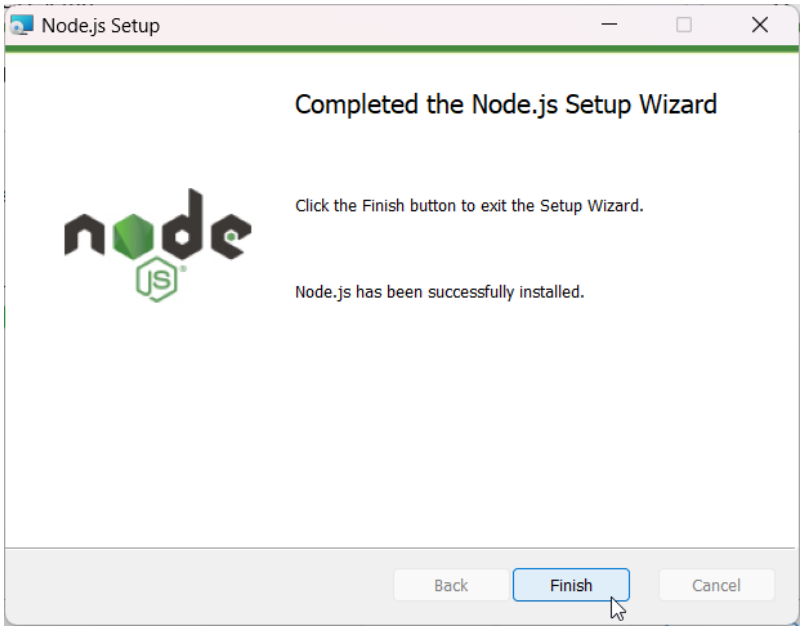
Figura 17:
Instalacion de NodeJs en Microsft Windows 11 Parte 7



Elborado por: Autor

Por último, la ventana final NodeJs cambia y presenta el mensaje que la instalación fue completada con éxito como en la imagen de la figura 18.

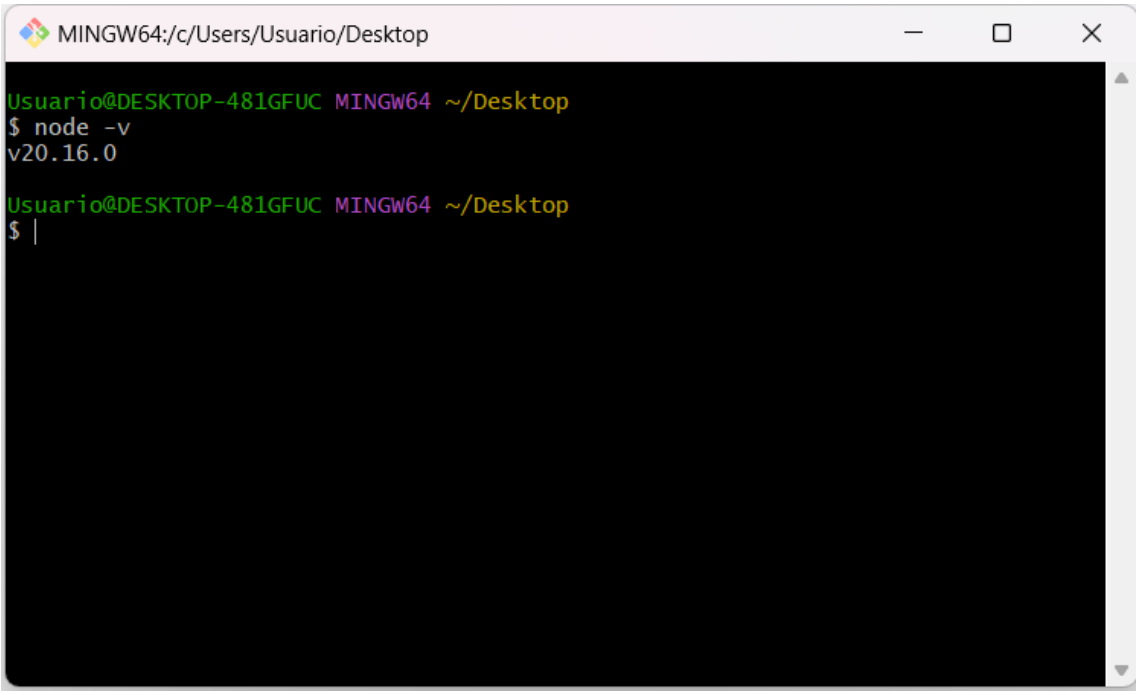
Figura 18:
Instalación completada con éxito.



Elaborado por: Autor

Por lo tanto, ahora desde nuestro sistema operativo Windows 11 podemos revisar que la instalación se ha realizado con éxito, nos dirigimos hacia una terminal en el ejecutamos el siguiente comando: “node -v” la terminal arrojara una respuesta indicándonos la versión que fue instalada NodeJs como en la imagen de la figura 19.

Figura 19 :
Verificación de la versión de NodeJs en la terminal.



Elaborado por: Autor

2.2.2 Vite: Plataforma de Desarrollo de Alto Rendimiento para Aplicaciones React

Vite: Es una herramienta de desarrollo web de última generación diseñada específicamente para optimizar el proceso de creación de aplicaciones web modernas, en particular aquellas basadas en frameworks como React, Vue entre otros. Su nombre, proveniente del francés "vite" (rápido), refleja su principal característica distintiva que sería la velocidad excepcionalmente alta para la creación del proyecto de desarrollo. (Wieruch, 2018)

¿Por qué Vite es una elección ideal para proyectos React?

Servidor de Desarrollo Basado en ESM: Vite emplea un servidor de desarrollo que aprovecha los módulos ECMAScript nativos, lo que permite un inicio casi instantáneo y actualizaciones en tiempo real de los componentes. Esta característica agiliza significativamente el ciclo de desarrollo de una aplicación, reduciendo los tiempos de espera y mejorando la productividad del desarrollador. (Osmani, 2017)

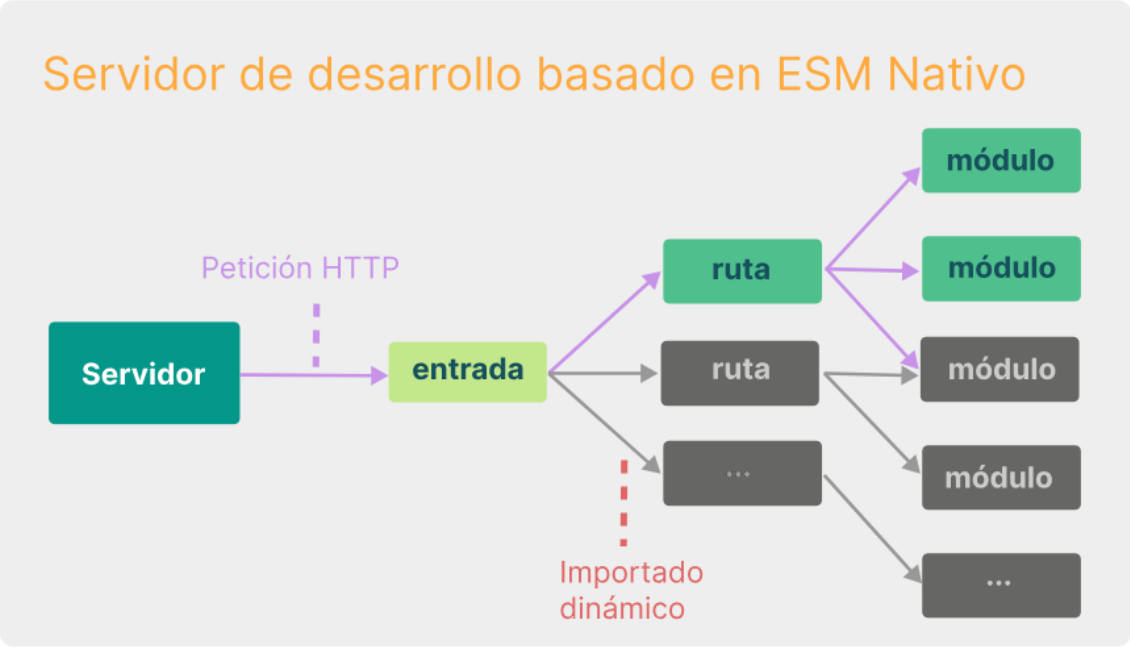
Hot Module Replacement (HMR): Vite implementa HMR de manera altamente eficiente, lo que significa que los cambios en el código fuente se reflejan en el navegador sin necesidad de una recarga completa de la página. Esta funcionalidad proporciona una experiencia de desarrollo más fluida y permite identificar y solucionar errores de manera más rápida.

Configuración Mínima: Vite ofrece una configuración predefinida que se adapta a la mayoría de los proyectos React, lo que permite a los desarrolladores comenzar a trabajar de inmediato sin tener que dedicar tiempo a configurar herramientas de compilación y empaquetado.

Soporte Nativo para TypeScript: Vite proporciona un soporte nativo para TypeScript, facilitando la creación de aplicaciones React escalables. TypeScript ayuda a prevenir errores comunes y mejora la mantenibilidad del código a largo plazo.

Optimización para Producción: Cuando se está listo para desplegar la aplicación en un entorno de producción, Vite genera paquetes de código optimizados para el rendimiento, asegurando una carga rápida y una experiencia de usuario óptima.

Figura 20:
Diagrama de Vite ESM Nativo



Fuente: https://es.vitejs.dev/guide/why

Vite sirve código fuente sobre ESM nativo. Básicamente, esto permitirá que el navegador se haga cargo de parte del trabajo de un empaquetador: Vite solo necesita transformar y servir código fuente a petición, según como lo solicite el navegador. El código detrás de las importaciones dinámicas condicionales sólo es procesado si realmente es usado en la pantalla actual.

Mecanismo de Funcionamiento:

Servidor de Desarrollo: Vite inicia un servidor que sirve los módulos de la aplicación de manera individual, lo que permite al navegador cargar solo los módulos que han cambiado, mejorando significativamente el tiempo de carga.

HMR: Vite utiliza HMR para actualizar los componentes de la aplicación en tiempo real sin necesidad de recargar toda la página.

Optimización para Producción: Vite genera un bundle de producción optimizado para el rendimiento, lo que garantiza una carga rápida de la aplicación en el navegador del usuario.

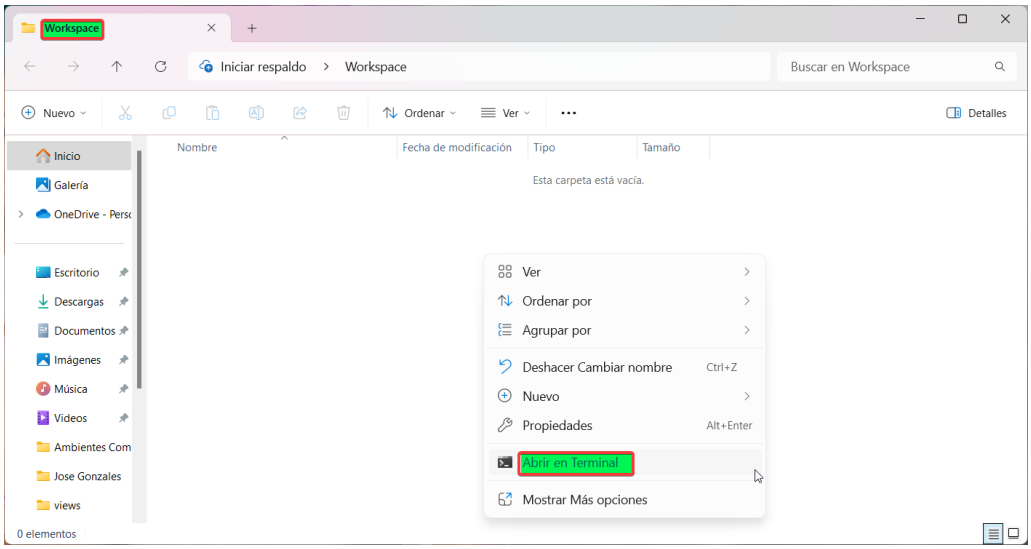
Por ende, Vite representa un avance significativo en el ámbito de las herramientas de desarrollo web, ofreciendo una experiencia de desarrollo rápida y eficiente para proyectos React gracias a su enfoque en la velocidad, la simplicidad de configuración y el soporte para las últimas tecnologías lo convierten en una elección popular entre los desarrolladores que buscan construir aplicaciones web modernas y escalables. (Chan, Fullstack React, 2018)

2.2.3 Creación de un nuevo proyecto React (Vite)

Ahora vamos a empezar con la creación de un proyecto pequeño de React en nuestro equipo de forma local, por ello se requiere que el lector de este documento ya tenga instalado los prerrequisitos como lo son el editor de código en mi caso personal utilizare Visual Studio Code el cual está en auge y además cuenta con herramientas que ayudan el desarrollo frontend como lo es una terminal embebida lo que permite trabajar de una mejor manera, a su vez utilizare Google Chrome como navegador principal ya que en el mismo puede instalarse herramientas de depuración de React las cuales nos van a permitir revisar nuestro trabajo. (Eisenman, React Quickly, 2016)

Ahora debemos crear una carpeta en el escritorio con un nombre claro que identifique que va a ser nuestro ambiente de trabajo en mi caso la creare como: “Workspace” podemos crearlo como se encuentra definido en la figura 21.

Figura 21:
Crear directorio de trabajo para React.

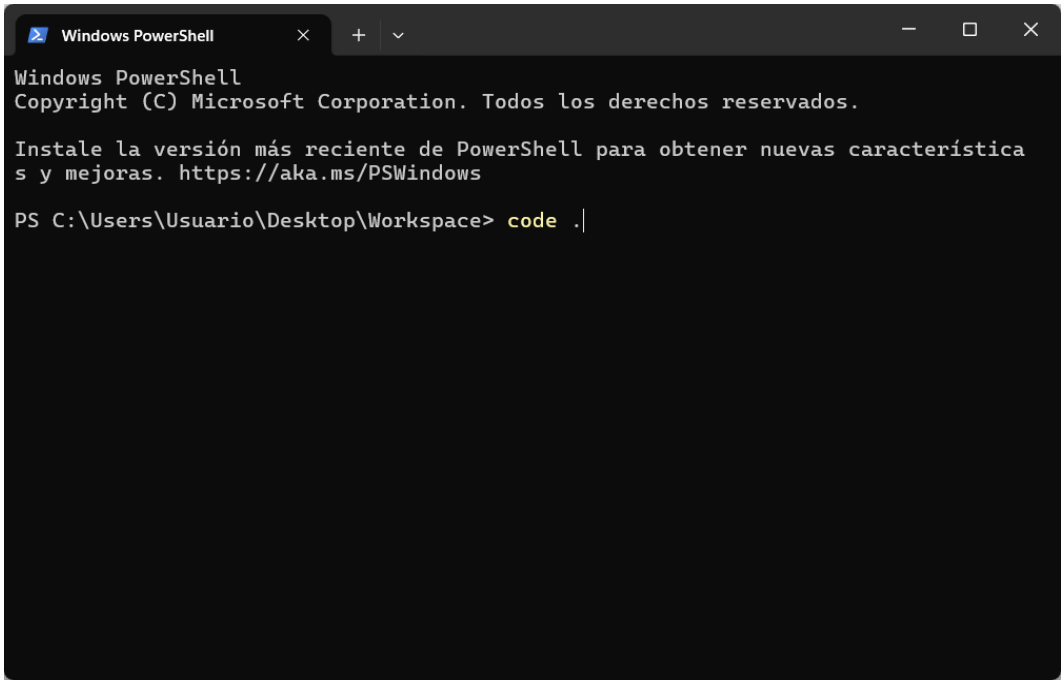


Elaborador por: Autor.

Una vez dentro del directorio que hemos creado abriremos una terminal de Windows y digitaremos el siguiente comando: “code .” lo que permitirá abrir VS Code de inmediato en el directorio de trabajo, así como se muestra en la figura 22.

De esta forma podemos ubicarnos rápido en la creación del proyecto, gracias a los atajos o teclas calientes proporcionadas por la herramienta permiten que el ingreso al editor de código sea mucho más amigable y eficaz.

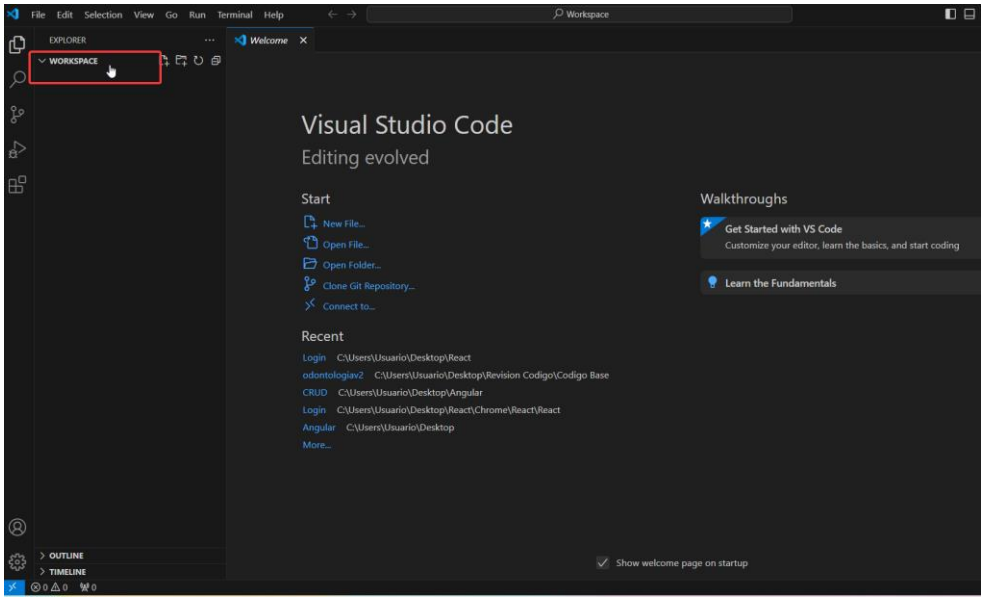
Figura 22:
Abrir VS Code mediante comandos



Elaborado por: Autor

Una vez ejecutado el siguiente comando: “code .” automáticamente VS Code se despliega en el directorio de trabajo, así como se muestra en la figura 23.

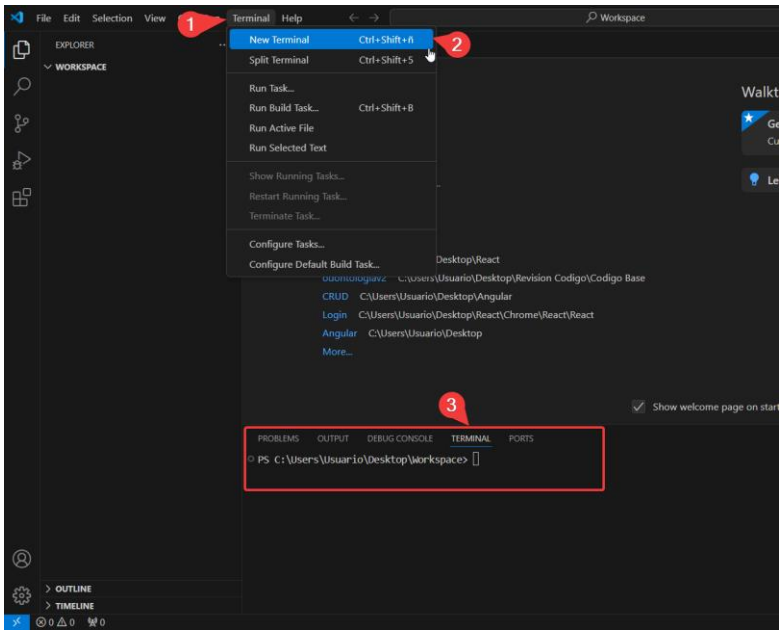
Figura 23:
Despliegue de VS Code en el directorio de Trabajo



Elaborador por: Autor

Ahora dentro de VS Code debemos abrir una terminal o línea de comandos embebida en nuestro editor de código podemos ver los pasos que se deben ejecutar para abrir la terminal, la misma se procederá a abrir en la parte inferior derecha de nuestro editor de codigo, así como se muestra en la imagen de la figura 24.

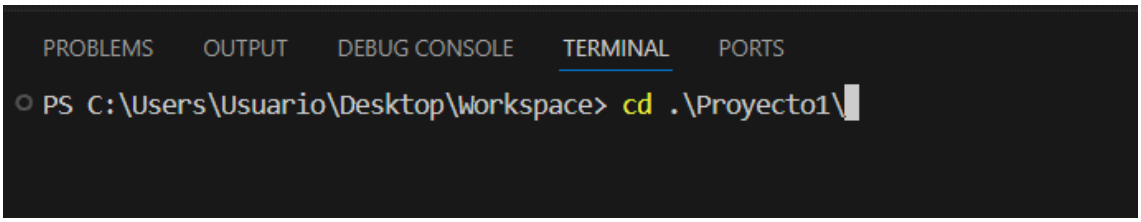
Figura 24:
Terminal embebida para trabajo VS Code



Elaborado por: Autor

Una vez tengamos abierta la terminal navegaremos hasta el directorio donde deseas crear el proyecto, en este caso en particular sea creado una carpeta en el navegador del VS Code para que dentro de ella se instalen todas las dependencias, como se muestra en la figura 25.

Figura 25:
Crear directorio del proyecto 7



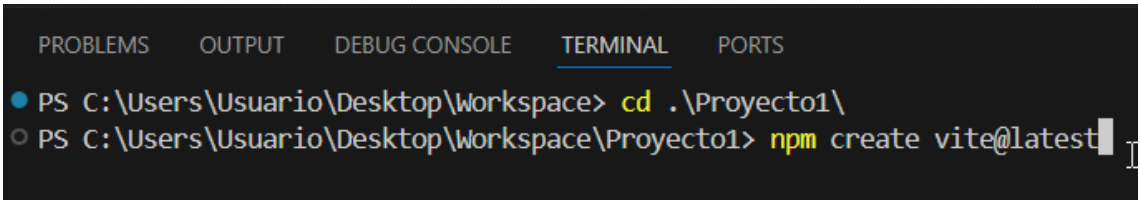
Elaborador por: Autor

Una vez dentro de la carpeta "Proyecto1" debemos Ejecuta el siguiente comando:

- ✓ npm create vite@latest

Esto a su vez desencadenara una serie de paso que veremos a continuación dentro de la creación del proyecto React-Vite, como se muestra en la figura 26.

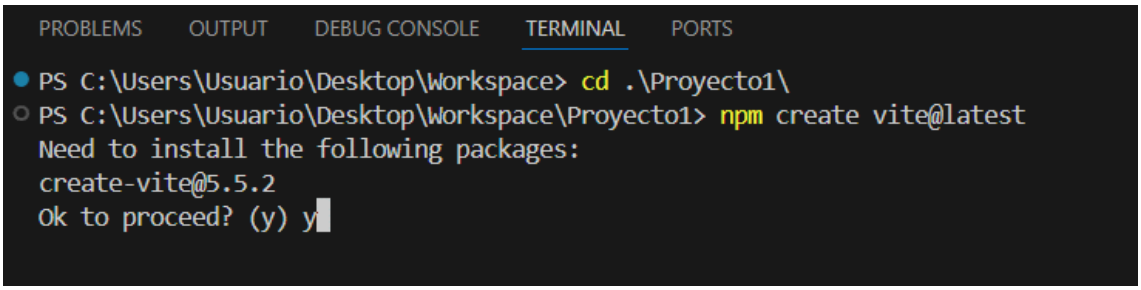
Figura 26:
Ejecución de comando de creación de proyecto React-Vite Parte 1



Elaborado por: Autor

Una vez ejecutado el programa de instalación de las dependencias del proyecto solicitara la instalación de paquetes necesarios para que React funcione correctamente por lo cual debemos digitar la letra "Y" en señal de aceptación de la descarga e instalación de dichos paquetes, como se indica en la figura 27.

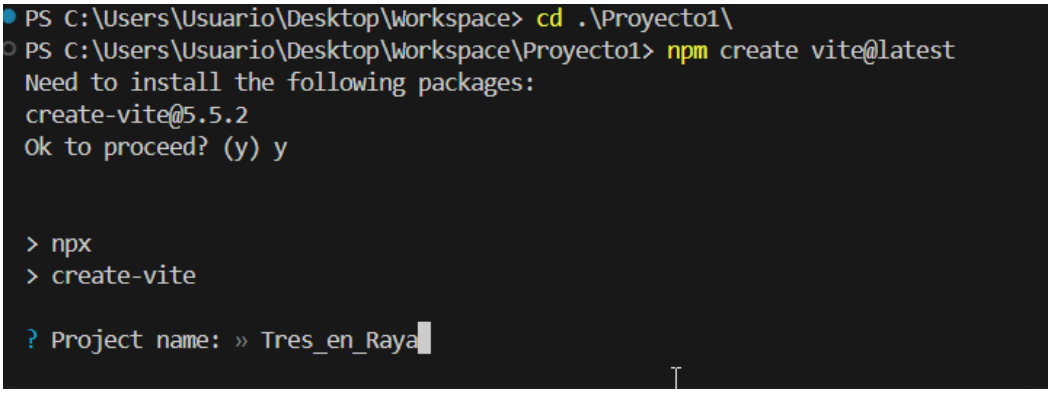
Figura 27:
Ejecución de comando de creación de proyecto React-Vite Parte 1



Elaborado por: Autor

Una vez aceptado la instalación de paquetes, debemos ingresar el nombre de nuestro proyecto, React por defecto te muestra un ejemplo de nombre con un placeholder que debe ser reemplazado por el nombre digitado de nuestro proyecto en este caso revisaremos el ejemplo de la web oficial de React del clásico juego Tres en raya, como se indica en la figura 28.

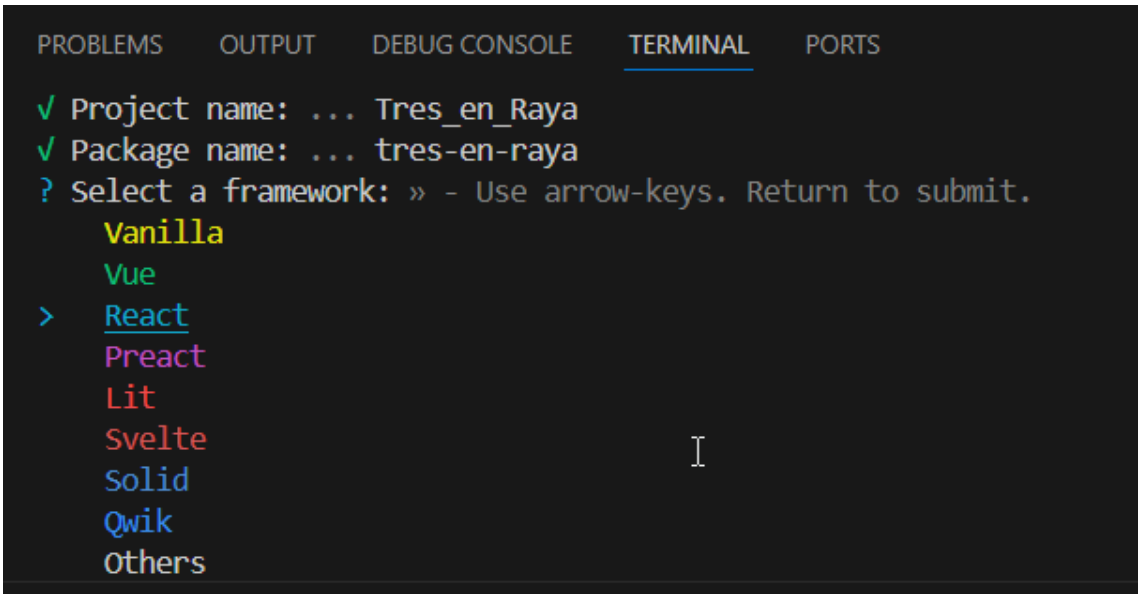
Figura 28:
Ingreso nombre del proyecto.



Elaborado por: Autor

Una vez ingresado el nombre de nuestro proyecto, React por defecto te muestra pondrá también un placeholder para crear el paquete que lo va a almacenar en este caso lo dejaremos igual al nombre del proyecto tres en raya. Ahora Vite solicita que le indiquemos con que librería/framework vamos a trabajar a su vez también podemos ver otros frameworks que fueron mencionados anteriormente, Por lo tanto, con nuestras teclas de dirección vamos a bajar a la tercera opción que es React, como se indica en la figura 29.

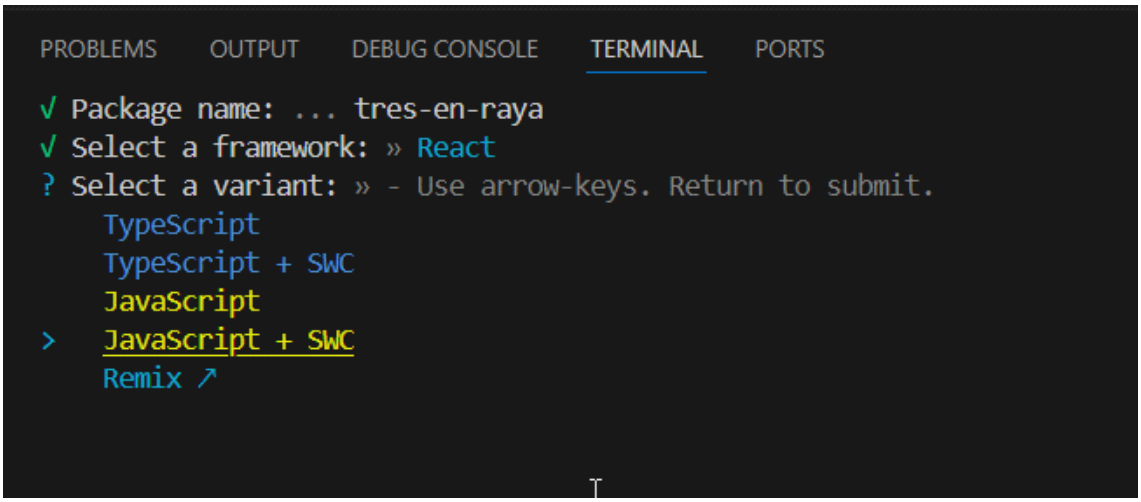
Figura 29:
Selección de Framework de Trabajo en Vite



Elaborado por: Autor

Una vez elegio el framework de trabajo React por defecto Vite te muestra las opciones en los lenguajes de programación que tiene en este caso elegiremos JavaScript pero como se mencionó anteriormente también sería oportuno que puedan elegir TypeScript siempre y cuando lo manejen correctamente la estructura es muy similar a JavaScript, SWC se refiere a un compilador JavaScript/TypeScript súper rápido que se utiliza en el proceso de construcción de aplicaciones web con Vite para mejorar el rendimiento y acelerar el tiempo de construcción de la aplicación, Por lo tanto con nuestras teclas de dirección vamos a bajar a la cuarta opción que es JavaScritp+SWC, como se indica en la figura 30. (Mardan, 2023)

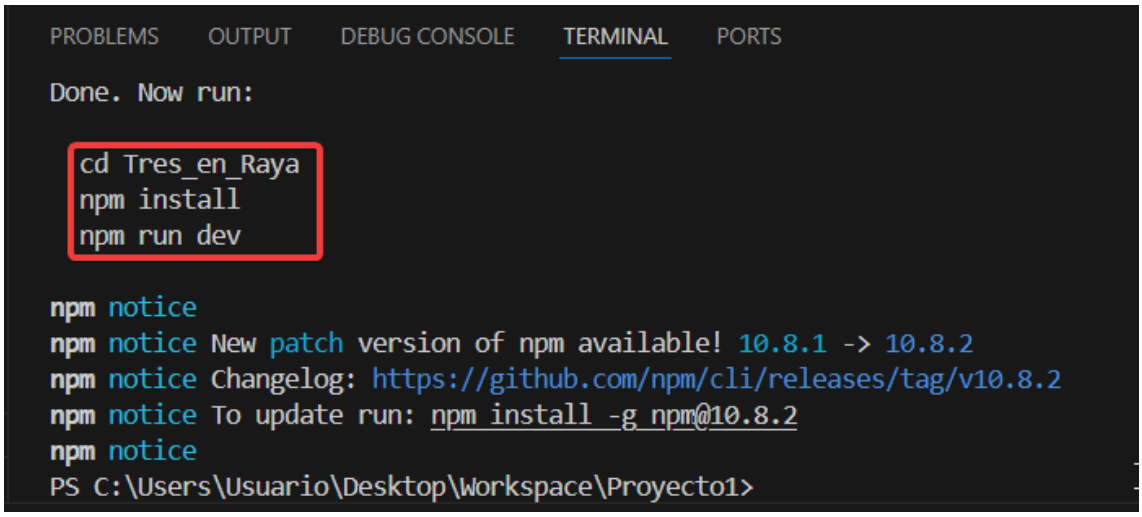
Figura 30:
Selección de Lenguaje de Programación JavaScript + SWC.



Elaborado por: Autor

Por último, damos un enter y el proceso de instalación de React JavaScript/TypeScript se ejecutará súper rápido, Luego Vite proporcionará los paso para el despliegue de nuestra aplicación que fue instalada con éxito, como se indica en la figura 31.

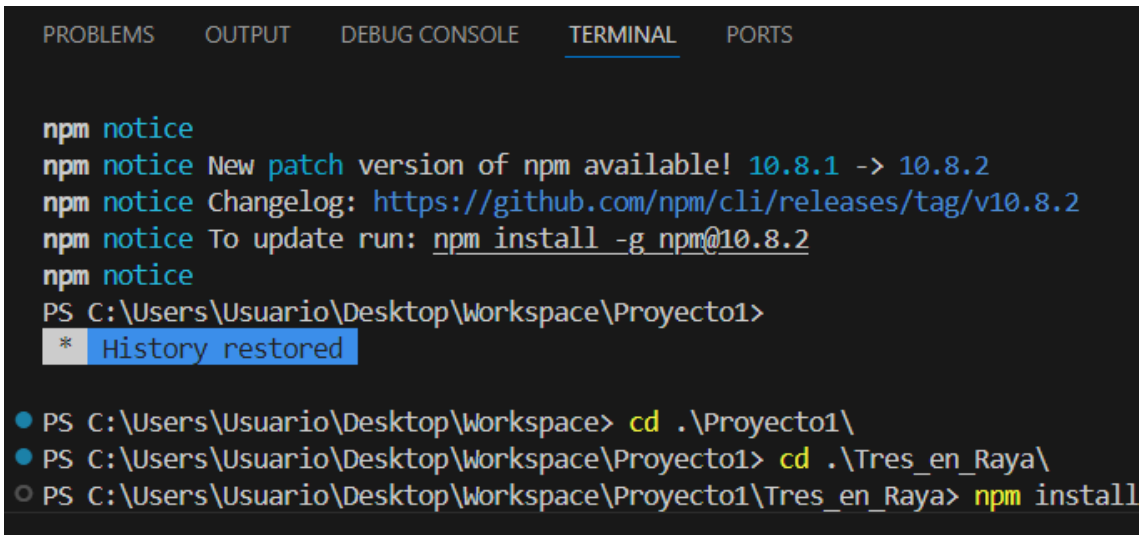
Figura 31:
Instalación de React-Vite completada.



Elaborador por: Autor

Para su despliegue debemos digitar las instrucciones que serian primero navegar hasta el proyecto tres en raya luego digitando el comando “npm install”, para poder levantar el proyecto instalado debemos ejecutar el comando “npm run dev”, así como se visualiza en las figuras 32.

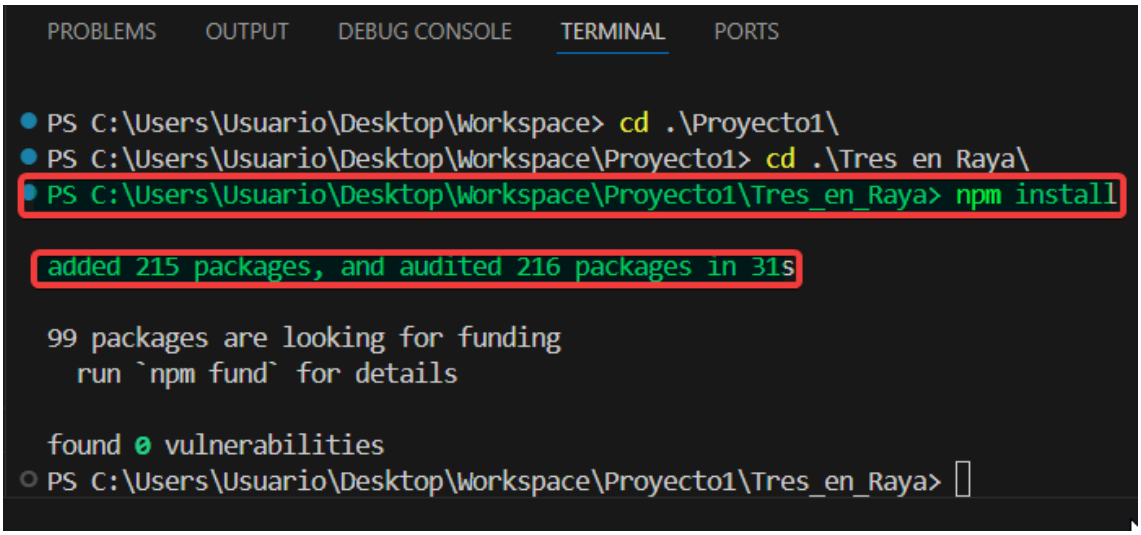
Figura 32:
Instalación de dependencias dentro de la carpeta del proyecto Tres en raya Parte 1



Elaborado por: Autor

Una vez ingresado el comando damas enter y tendremos que esperar un tiempo a que se realice la ejecución exitosa del comando npm install dentro del directorio principal del proyecto, así como se visualiza en la figura 33.

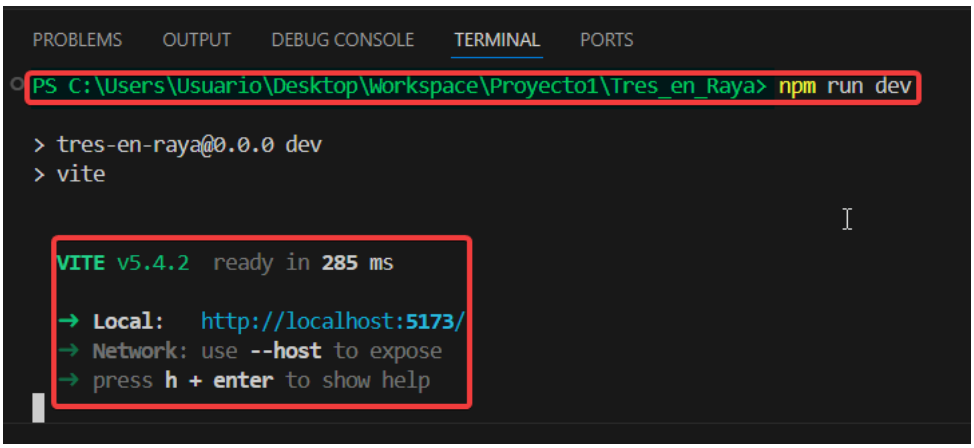
Figura 33:
Instalación de dependencias dentro de la carpeta del proyecto Tres en raya Parte 2



Elaborado por: Autor

Una vez realizada la instalación de las dependencias dentro de la misma carpeta del proyecto debemos levantar el servicio para poder visualizarlo en un navegador, para ello debemos ejecutar el comando: "npm run dev" como se visualizar en la figura 34.

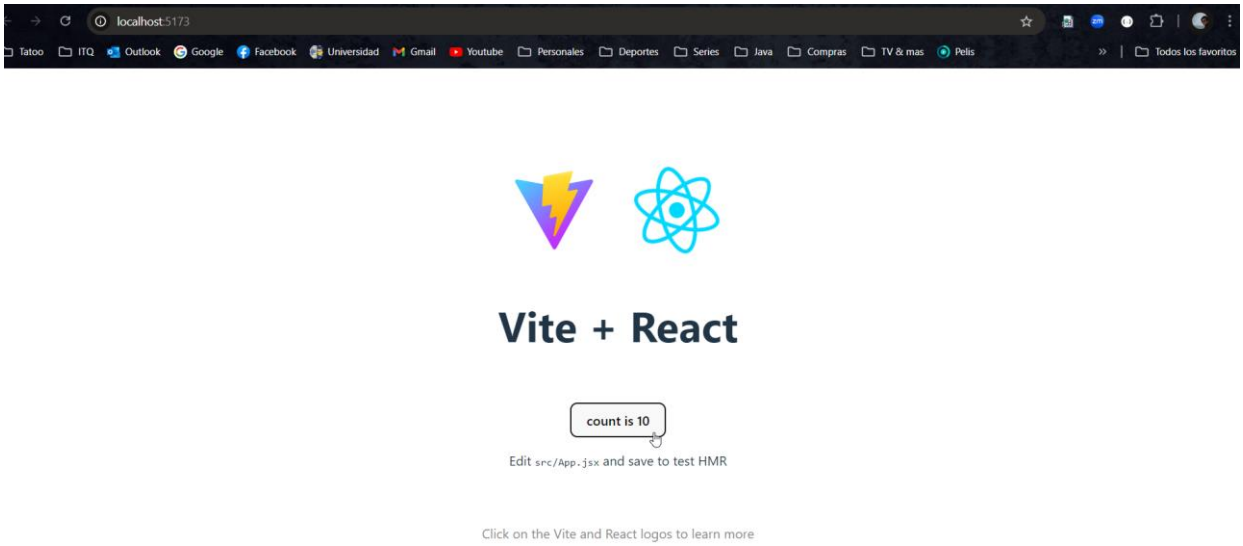
Figura 34:
Despliegue de la Aplicación React-Vite instalada anteriormente.



Elaborado por: Autor

Ahora podremos divisar la página por defecto que se ha creado con la instalación que realizamos en los pasos anteriores, para ello levantar el servicio es primordial con el comando: “npm run dev”, debemos posicionarnos en la opción de. “Local: http//localhost:5173” que es la URL en la cual se levantó el servicio y para poder visualizarlo en un navegador vamos a dar clic en la tecla “CTRL” + clic sobre esta opción y se abrirá nuestro navegador como se visualizar en la figura 35. (Drasner, 2022)

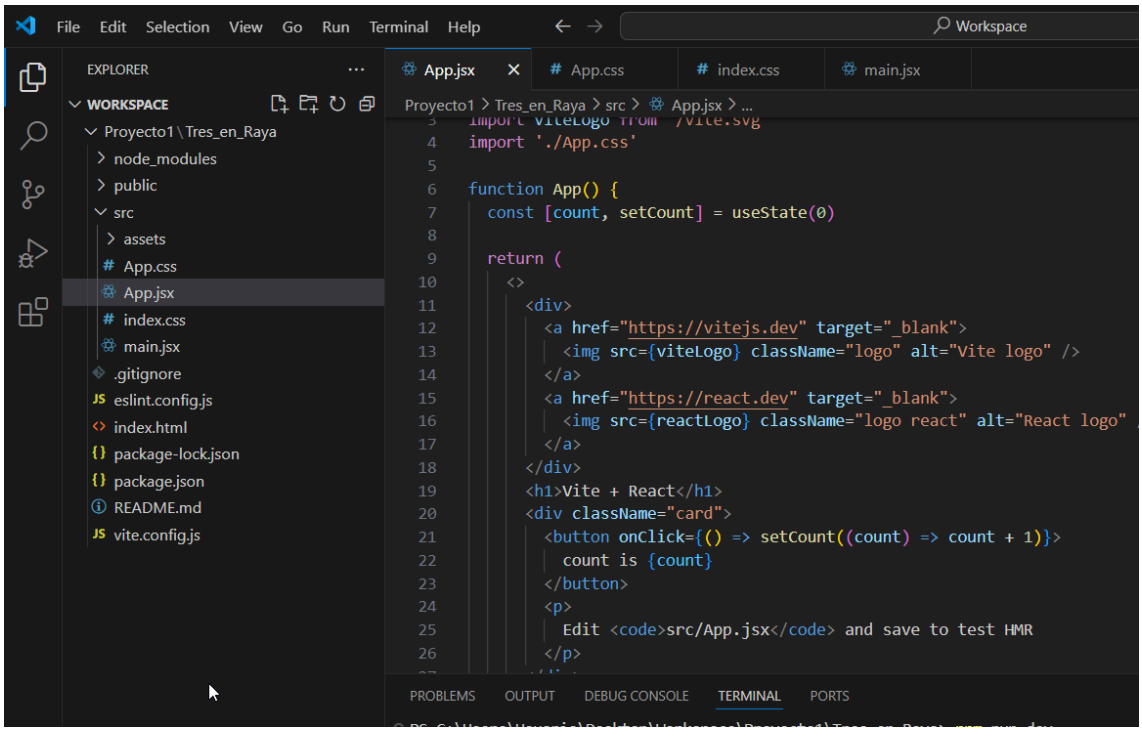
Figura 35:
Visualización del Despliegue de la Aplicación en Google Chrome



Elaborado por: Autor

Ahora podremos divisar el proyecto en su totalidad al tener instalado ya y funcional nuestro proyecto podremos crear componentes o editarlo a nuestra conveniencia ahora hay que remarcar que los archivos de edición son “App.jsx” que es aquel archivo que cuenta con las funciones para llamar a los elementos de JS, dentro de la estructura podemos divisar una carpeta node_modules esta carpeta es fundamental dado que de ella parte para tomar las dependencias de node y si el proyecto cambia de equipo o se encuentra en un repositorio debe estar en el archivo git ignore dado que esta carpeta se crea cuando hemos ejecutado el comando: ” npm install ” , a su vez también tenemos archivos de diseño como son las hojas en cascada representada en el index.css así como se muestra en la figura 36.

Figura 36:
Estructura del proyecto.



Elaborado por: Autor

A continuación, presentaremos un ejemplo para realizar el juego de tres en raya el mismo estará disponible el siguiente capítulo de este libro así mismo en la web oficial de React, la aplicación de React tiene como principal archivo a “App.jsx”, En el cual se procederá a efectuar los siguientes cambios, Ahora bien antes de continuar con en este apartado empezaremos a explicar los conceptos a través de React que facilitan la creación de componentes , la inmutabilidad de los mismo, el manejo de propiedades y estados de cada componente.

Esto también conlleva a explicar la estructura que maneja React dentro de sus directorios y cómo podemos utilizarlos en beneficio nuestro para la creación de aplicaciones que sean escalables y ordenadas. A medida que vamos avanzando con estos pasos indicaremos la resolución del ejercicio planteado.

2.3 Unidades Fundamentales de la Construcción de Interfaces de Usuario Componentes, Props, Estado y Hooks en React

2.3.1 Componentes en React:

En el paradigma de programación de React, los **componentes** se erigen como los bloques constructores fundamentales para la creación de interfaces de usuario interactivas y escalables. Estos elementos encapsulan tanto la estructura (JSX) como la

lógica (JavaScript) necesaria para representar una porción específica de la interfaz (Facebook., 2017)

Características Distintivas de los Componentes en React:

- **Reutilización:** Los componentes son diseñados para ser reutilizados en múltiples partes de una aplicación, promoviendo la consistencia y reduciendo la duplicación de código.
- **Encapsulación:** Cada componente mantiene un estado interno y una lógica propia, lo que favorece su modularidad y la facilidad de mantenimiento.
- **Jerarquía:** Los componentes pueden anidarse de manera jerárquica, creando una estructura de árbol que refleja la composición de la interfaz de usuario.
- **Declaración:** La sintaxis JSX, empleada en React, permite describir la interfaz de usuario de forma declarativa, declarando lo que se desea mostrar en pantalla en lugar de como se debe lograr.

Tipos de Componentes:

- **Componentes Funcionales:** Son funciones de JavaScript puras que reciben props como entrada y devuelven JSX. Son ideales para componentes simples y sin estado interno.
- **Componentes de Clase:** Basados en la clase React-Component, ofrecen un mayor control sobre el ciclo de vida del componente y la gestión del estado. Sin embargo, con la introducción de los hooks, su uso se ha vuelto menos frecuente para componentes simples. (Quirk, 2021)

2.3.2 Props

A breves rasgos se los puede describir como la transmisión de datos entre componentes en React, la palabra props viene de un diminutivo de propiedades en si los props (propiedades) sirven como mecanismo para pasar datos de un componente padre a un componente hijo.

Actúan como atributos personalizados que se asignan a un componente. Los props son inmutables, es decir, no pueden ser modificados dentro del componente hijo. Esta unidireccionalidad de flujo de datos es fundamental para mantener la predictibilidad y la facilidad de razonamiento en las aplicaciones React.

2.3.3 Estado

Es la gestión de datos internos de un componente es decir es el estado en React representa los datos internos de un componente que pueden cambiar con el tiempo y provocar renderizados. Cuando el estado de un componente cambia, React actualiza de manera eficiente la interfaz de usuario para reflejar los nuevos datos. El hook useState es la herramienta principal para gestionar el estado en componentes funcionales.

2.3.4 Hooks

Se comprende como un extendiendo las funcionalidades de los componentes funcionales es decir los hooks son funciones que permiten "enganchar" (hook) tus componentes funcionales a características de React.

Estos hooks proporcionan una forma concisa y declarativa de gestionar estado, efectos secundarios y otras funcionalidades que antes estaban limitadas a las clases. El hook useState es uno de los más básicos y se utiliza para agregar estado a un componente funcional.

Ejemplo en JavaScript del useState:

```
import { useState } from 'react';
function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Contador: {count}</p>
      <button onClick={() => setCount(count + 1)}>Incrementar</button>

    </div>
  );
}
```

En el ejemplo se puede ver que el useState devuelve un arreglo con dos elementos:

- ✓ count: El valor actual del estado.
- ✓ setCount: Una función para actualizar el valor del estado.

Cada vez que se hace clic en el botón, setCount se llama con un nuevo valor, lo que provoca que el componente se renderice con el contador actualizado.

Resumen del Capítulo 2

En el segundo capítulo de este libro hemos visto la importancia de la instalación de los prerrequisitos para los proyectos de la librería/framework React , lo cual nos permitió tener un equipo actualizado además hemos revisado las principales estructuras, primero hemos instalado NodeJs uno de los temas principales para que React pueda funcionar correctamente para optimizar su funcionamiento también hemos instalado los paquetes dependientes del proceso como lo es NPM el cual permite interactuar con la terminal e instalar las dependencias para la creación de los proyectos de diseño y creación de aplicaciones multiplataforma, hemos detallado paso a paso la instalación de NodeJs, una vez realizada la instalación de Node hemos necesitado de un editor de texto lo cual nos va a permitir generar la instalación de las dependencias en nuestro ambiente de trabajo, se presenta paso a paso la creación de un proyecto React-Vite, ya que Vite sirve código fuente sobre ESM nativo. Básicamente, esto permitirá que el navegador se haga cargo de parte del trabajo de un empaquetador. Posterior a esto hemos realizado un código de ejemplo de cada estructura que maneja React.

En si se puede considerar lo siguiente en cuanto a las estructuras que maneja React los props son mecanismo de paso de datos de un componente padre a un hijo, unidireccional e inmutable. Mientras que los estados son datos internos de un componente que pueden cambiar y causar renderizados. Hay que recordar que existen otros hooks como lo son useEffect para efectos secundarios, useContext para compartir datos a través de componentes, useReducer para gestionar estados complejos (este último puede detallarse con redux, pero dependerá de su complejidad y utilidad) y useState es un hook para gestionar el estado en componentes funcionales. La elección entre props y estado depende de si los datos son proporcionados desde fuera del componente (props) o si son gestionados internamente (estado). Comprender estos conceptos es fundamental para construir aplicaciones React eficientes y mantenibles.

Hay que tener en cuentan las siguientes consideraciones adicionales las versiones que se utilizan es decir asegúrate de utilizar versiones estables y actualizadas de Node.js, npm y tu editor de código para garantizar un entorno de desarrollo óptimo. Recordar la configuración se encarga de personalizar tu editor de código con extensiones y temas que se adapten a tu flujo de trabajo lo que permite un aprendizaje continuo y para esto mantenerse actualizado con las últimas novedades y mejores prácticas en el desarrollo frontend es fundamental ya que al cumplir con estos prerrequisitos, estarás dando un salto de calidad para comenzar a construir aplicaciones web interactivas y escalables utilizando React.

Capítulo 3

Programando en React una revisión de su modelo de trabajo.

En React es necesario crear componentes los mismo que son parte del proceso de elaboración de la web que deseamos realizar, ahora bien, estos componentes trabajan en un lenguaje de programación ya muy conocido como lo es JavaScript o no tanto porque también podemos trabajar con TypeScript todos estos conceptos y manejo de nuestra librería/framework lo veremos a continuación. (Lonsdorf, 2023)

3.1 Normas y Estructura Fundamentales en el Desarrollo con React

React, una biblioteca JavaScript de código abierto, ha revolucionado la forma en que construimos interfaces de usuario. Su enfoque declarativo y basado en componentes ha establecido estándares de eficiencia y mantenibilidad en el desarrollo frontend.

3.1.1 Normas Esenciales en la Programación con React

Esto es considerado como la declaración en nuestra herramienta de trabajo lo más importante para React son los componentes como funciones, es decir, los componentes se conciben como funciones puras que reciben entradas (props) y devuelven un árbol de elementos JSX que representa la interfaz de usuario. (Khourshid, 2022)

JSX: Una sintaxis similar a HTML que permite escribir estructuras de UI de forma concisa y legible su composición es

- ✓ **Jerarquía de componentes**: Los componentes se pueden anidar para crear interfaces más complejas.
- ✓ **Reutilización**: Los componentes bien diseñados son reutilizables en diferentes partes de la aplicación, promoviendo su modularidad y la consistencia.

Estado:

- ✓ **Inmutabilidad**: El estado de un componente debe ser tratado como inmutable. Para modificarlo, se utiliza el método setState.
- ✓ **Flujo de datos unidireccional**: La información fluye de forma unidireccional, desde los componentes padres a los hijos, a través de las props.

Ciclo de vida:

- ✓ **Métodos del ciclo de vida**: Estos métodos permiten ejecutar código en diferentes etapas del ciclo de vida de un componente, como al montar, actualizar o desmontar. (Florence, 2023)

3.2 Estructura Típica de un Proyecto React

Un proyecto React suele organizarse de la siguiente manera, aquí podemos detallar grosso modo todos los elementos que se instalaron en el capítulo anterior:

Tabla 2:
Componentes principales de un proyecto de React

Elemento de la carpeta del Proyecto	Definición
public	Contiene archivos estáticos como index.html que sirve como punto de entrada de la aplicación
src	Carpeta destinada a almacenamiento de la funcionalidad de la aplicación
index.js	El punto de entrada principal donde se renderiza el componente raíz.
App.js	El componente raíz de la aplicación, que encapsula toda la lógica y estructura de la UI.
componentes	Contiene componentes reutilizables, organizados por funcionalidad o tipo (e.g., Button, Header, Form).
styles	Contiene archivos de estilos (CSS, Sass, Less, etc.).
assets	Almacena imágenes, fuentes y otros activos estáticos

Ejemplo de un Componente Funcional Simple

JavaScript

```
import React from 'react';

function Greeting(props) {
  return <h1>Hola, {props.name}!</h1>;
}

export default Greeting;
```

Tener en cuenta las siguientes consideraciones adicionales para la creación y desarrollo de aplicaciones en React se necesita una gestión de estado esto se aplica para aplicaciones más complejas, se pueden utilizar librerías como Redux o Context API para gestionar el estado global también cabe mencionar que el enrutamiento no esta presente en React como tal así que debemos utilizar librerías como React-Router que permiten crear aplicaciones de una sola página con múltiples vistas.

Es fundamental escribir pruebas unitarias y de integración para garantizar la calidad del código. Utilice herramientas de desarrollo como React DevTools para inspeccionar y depurar sus componentes. En resumen, React proporciona una base sólida para construir interfaces de usuario modernas y escalables. Al comprender los principios fundamentales y siguiendo las mejores prácticas, podrá crear aplicaciones web de alta calidad. Ahora vamos a crear o editar nuestra clase App.jsx la misma debe ahora tener la facultad para almacenar el siguiente enunciado:

Crear el juego de tres en raya en React utilizando un contador que permita retomar la partida en pasos anteriores debe utilizar el diseño de un array de 3 x 3 en el cual se permita solventar el juego, a su vez debe declarar el ganador del juego cuando este complete sus 3 aciertos.

Este ejemplo no asume ningún conocimiento previo de React. Las técnicas que aprenderás en el ejemplo son fundamentales para crear cualquier aplicación de React, y comprenderlas por completo te dará una comprensión profunda de React.

El código en App.js crea un *componente*. En React, un componente es una pieza de código reutilizable que representa una parte de una interfaz de usuario. Los componentes se utilizan para representar, administrar y actualizar los elementos de la interfaz de usuario en su aplicación. Miremos la componente línea por línea en la figura 37 y veamos qué está pasando:

Figura 37:
Creación del Componente elaboración App.js parte 1

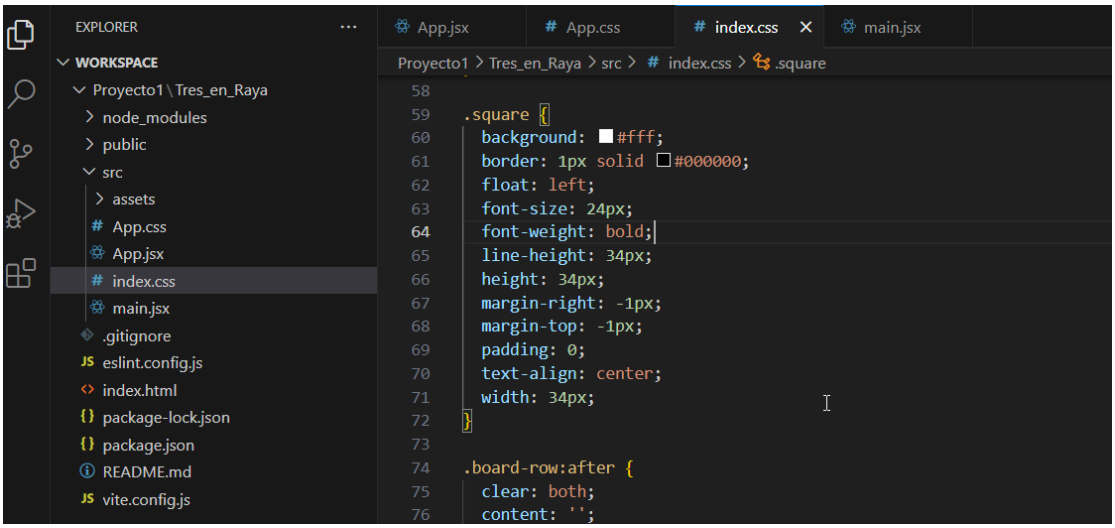
```
export default function Square() {
  return <button className="square">X</button>;
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe#overview

La primera línea define una función llamada Square. La palabra clave de JavaScript export hace que esta función sea accesible fuera de este archivo. La palabra clave default les dice a otros archivos que usan su código que es la función principal en su archivo.

La segunda línea devuelve un botón. La palabra clave de JavaScript return significa que lo que viene después se devuelve como un valor a la persona que llama a la función. <button> es un *elemento JSX*. Un elemento JSX es una combinación de código JavaScript y etiquetas HTML que describe lo que te gustaría mostrar. className="square" es una propiedad de botón o *prop* que le dice a CSS cómo diseñar el botón. X es el texto que se muestra dentro del botón y </button> cierra el elemento JSX para indicar que ningún contenido siguiente debe colocarse dentro del botón.

Figura 38:
Definición de Hojas de estilo en cascada



Elaborado por: Autor

Haz doble clic en el archivo llamado styles.css en la sección de la carpeta workspace creada en el capítulo 2. Este archivo define los estilos para tu aplicación React. Los primeros dos *selectores CSS* (* y body) definen el estilo de grandes partes de su aplicación, mientras que el selector ".square" define el estilo de cualquier componente donde la propiedad className está establecida en square. En tu código, eso coincidiría con el botón de tu componente Square en el archivo App.js como se visualiza en la figura 38.

Haz clic en el archivo llamado index.js en la de la carpeta workspace creada en el capítulo 2. No debemos editar este archivo durante el ejemplo, pero es este el puente entre el componente que creaste en el archivo App.js y el navegador web.

A continuación revisaremos como pasar datos a través de props , querrás cambiar el valor de un cuadrado de vacío a «X» cuando el usuario haga clic en el cuadrado. Con la forma en que ha construido el tablero hasta ahora, necesitarías copiar y pegar el código que actualiza el cuadrado nueve veces (¡una vez por cada cuadrado que tengas)! En lugar de copiar y pegar, la arquitectura de componentes de React te permite crear un componente reutilizable para evitar el código duplicado desordenado. (React Dev, 2024)

Figura 39:
Definición de la función para el cuadrado del juego.

```
function Square() {
  return <button className="square">1</button>;
}

export default function Board() {
  // ...
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe#overview

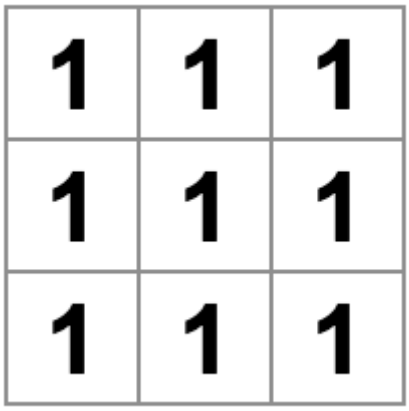
Primero, ve a copiar la línea que define el primer cuadrado (<button className="square">1</button>) de tu componente Board en un nuevo componente Square, y Luego, actualiza el componente Board para renderizar ese componente Square usando la sintaxis JSX de tal forma que puedas visualizarlo varias a la postre debemos observa cómo, a diferencia de los divs del navegador, tus propios componentes Board y Square deben comenzar con una letra mayúscula.

Figura 40:
Definición de función Board

```
// ...
export default function Board() {
  return (
    <>
      <div className="board-row">
        <Square />
        <Square />
        <Square />
      </div>
      <div className="board-row">
        <Square />
        <Square />
        <Square />
      </div>
      <div className="board-row">
        <Square />
        <Square />
        <Square />
      </div>
    </>
  );
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe#overview

Vamos a ver que el resultado arrojado en el tablero seria el siguiente:



Ahora cada cuadrado dice «1». Para arreglar esto, utilizaremos las *props* para pasar el valor que debe tener cada cuadrado del componente principal (Board) al componente secundario (Square). Por ello actualizaremos el componente Square para leer la propiedad "value" o valor que pasarás desde el Tablero, hay que definir la function Square({ value }) indica que al componente Square se le puede pasar un objeto llamado

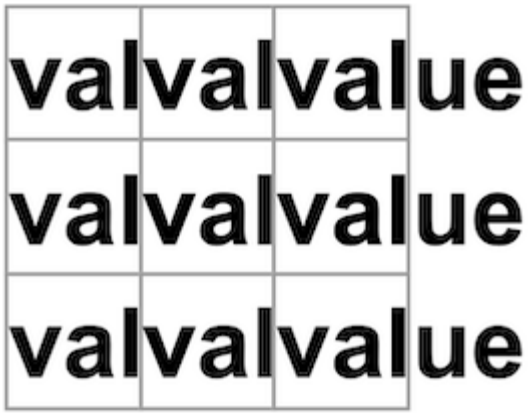
value. Ahora deseas mostrar ese value en lugar de 1 dentro de cada cuadrado. Intenta hacerlo así:

Figura 41:
Función Square con parámetros (Props)

```
function Square({ value }) {
  return <button className="square">value</button>;
}
```

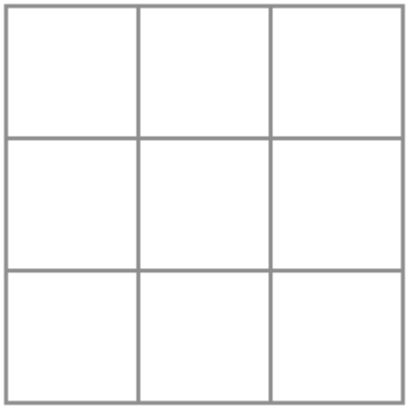
Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Vaya, este no es el resultado que esperábamos:



Necesitamos representar la variable de JavaScript llamada value de nuestro componente, no la palabra «valor». Para «escapar a JavaScript» desde JSX, necesitas llaves. Agrega llaves alrededor de value en JSX de esta manera:

Por ahora, deberías ver un tablero vacío:



Esto se debe a que el componente Board aún no ha pasado la prop value a cada componente Square que representa. Para solucionarlo, agrega el complemento value a cada componente Square representado por el componente Board

Figura 42:
Representación función Board aplicando el valor en cada función.

```
export default function Board() {
  return (
    <>
      <div className="board-row">
        <Square value="1" />
        <Square value="2" />
        <Square value="3" />
      </div>
      <div className="board-row">
        <Square value="4" />
        <Square value="5" />
        <Square value="6" />
      </div>
      <div className="board-row">
        <Square value="7" />
        <Square value="8" />
        <Square value="9" />
      </div>
    </>
  );
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Ahora deberías ver una cuadrícula de números nuevamente:

1	2	3
4	5	6
7	8	9

Rellenemos el componente Square con una X al hacer clic en él. Se declara una función llamada handleClick dentro del Square. Luego, agrega onClick a las props del elemento <button> JSX devuelto por el componente Square esto nos permitirá crear un un componente interactivo que genera un registro que dice "¡hiciste clic!" en la pestaña Consola en el Navegador de Chrome en la pestaña consola cuando utilizas el botón f12 Al hacer clic en el cuadrado más de una vez, se registrará "¡hiciste clic!" de nuevo. Los registros repetidos en la consola con el mismo mensaje no crearán más líneas en la consola.

Figura 43:
Función handClick para presentar datos.

```
function Square({ value }) {
  function handleClick() {
    console.log('¡hiciste clic!');
  }

  return (
    <button
      className="square"
      onClick={handleClick}
    >
      {value}
    </button>
  );
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Como siguiente paso, deseas que el componente Square «recuerde» que hiciste clic y lo rellene con una marca «X». Para «recordar» cosas, los componentes usan estado.

React proporciona una función especial llamada useState que puedes llamar desde tu componente para permitirle «recordar» cosas. Almacenemos el valor actual del Square en el estado, y cambiémoslo cuando se haga clic en Square.

Figura 44:
Declaración de useState para recordar valores.

```
import { useState } from 'react';

function Square() {
  const [value, setValue] = useState(null);

  function handleClick() {
    //...
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Importa useState en la parte superior del archivo. Elimina la propiedad value del componente Square. En su lugar, agrega una nueva línea al comienzo del componente Square que llame a useState. Haz que este devuelva una variable de estado llamada value.

Value se encarga de almacenar el valor y setValue es una función que se puede usar para cambiar el valor. El null pasado a useState se usa como valor inicial para esta variable de estado, por lo que value aquí comienza igual a null. Dado que el componente Square ya no acepta props, elimina la prop value de los nueve componentes Square creados por el componente Board.

Figura 45:
Funciones Square retirando Value para pasar por Props

```
// ...
export default function Board() {
  return (
    <>
      <div className="board-row">
        <Square />
        <Square />
        <Square />
      </div>
      <div className="board-row">
        <Square />
        <Square />
        <Square />
      </div>
      <div className="board-row">
        <Square />
        <Square />
        <Square />
      </div>
    </>
  );
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe#overview

Ahora cambia Square para mostrar una «X» cuando se haga clic. Reemplaza el controlador de evento console.log ("¡hiciste clic!"); con setValue('X');. Ahora aplicando nuestro conocimiento de setValue en los componentes entonces en nuestro componente Square se verá así:

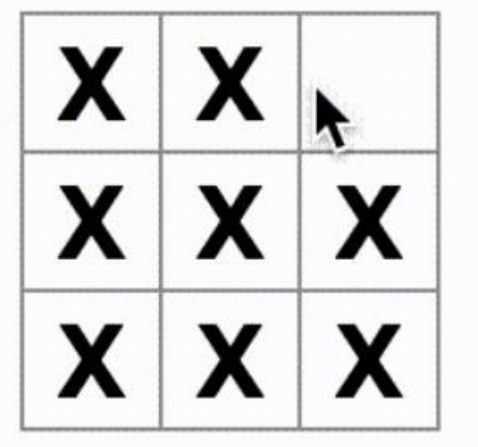
Figura 46:
Componente enviando X para digitar en tablero.

```
function Square() {
  const [value, setValue] = useState(null);

  function handleClick() {
    setValue('X');
  }

  return (
    <button
      className="square"
      onClick={handleClick}
    >
      {value}
    </button>
  );
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe#overview

Al llamar a esta función set desde un controlador onClick, le estás diciendo a React que vuelva a renderizar ese Square cada vez que se hagas clic en <button>. Después de la actualización, el value del Square será 'X', por lo que verás la «X» en el tablero de juego.

Si haces clic en cualquier cuadrado, debería aparecer una «X»:

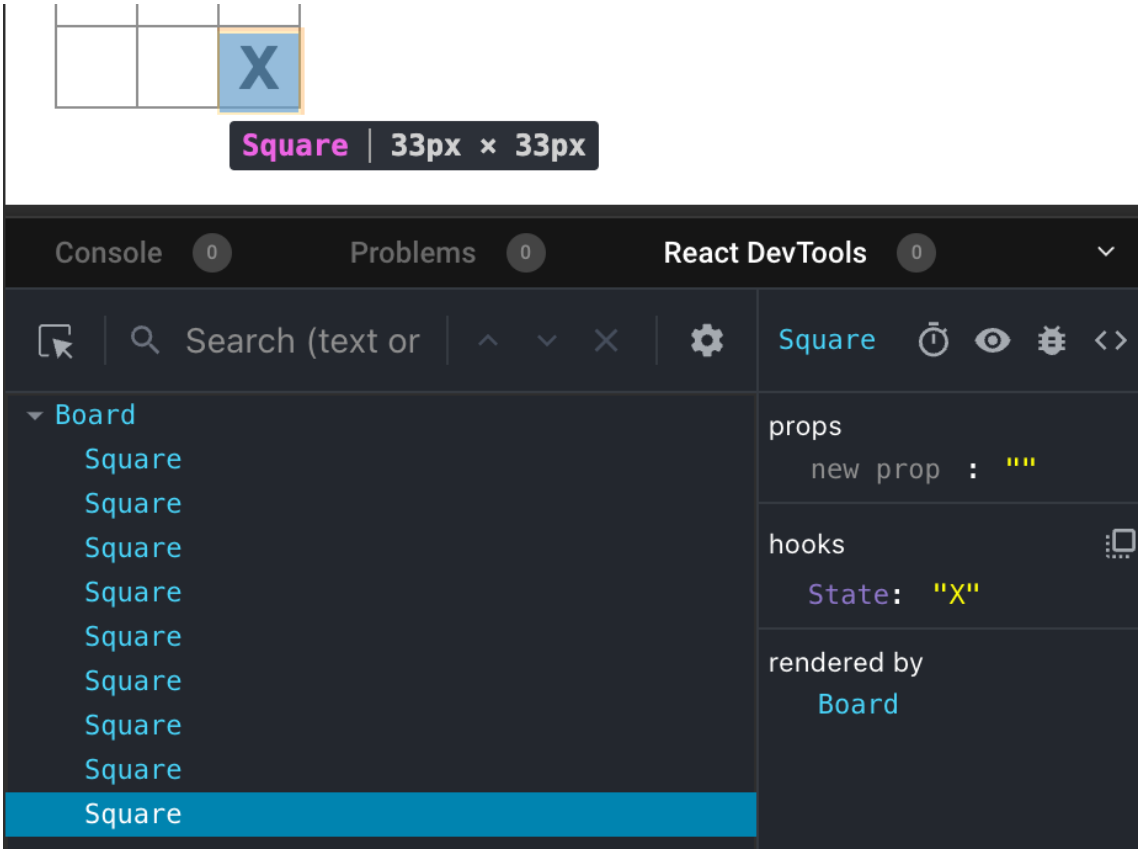


Ten en cuenta que cada Square tiene su propio estado: el value almacenado en cada Square es completamente independiente de los demás. Cuando llamas a una función set en un componente, React también actualiza automáticamente los componentes secundarios dentro del mismo componente.

3.3 React Developer Tools

React DevTools es una extensión que nos permite verificar las props y el estado de tus componentes React. Puedes encontrar la pestaña React DevTools para el desarrollo local, React DevTools está disponible tanto en Chrome, Firefox, y Edge extensión del navegador. Después de instalarlo, la pestaña *Componentes* aparecerá en las Herramientas de desarrollo de tu navegador para los sitios que utilizan React, Es decir que una vez instalado debes ubicarte con la opción f12 en la sección de componentes.

Figura 47:
React Dev Tools en Navegador



Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Para inspeccionar un componente en particular en la pantalla, usa el botón en la esquina superior izquierda de React Dev Tools, aquí podremos divisar los diferentes estados y propiedades (props) a su vez esto permite que revisemos de manera exhaustiva los valores que pasan o cambian en cada elemento de React.

Ahora conecta la prop onSquareClick a una función en el componente Board que llamarás handleClick. Para conectar onSquareClick a handleClick, pasa una función a la prop onSquareClick del primer componente Square

Figura 48:
Función Board con Array de asignación

```
export default function Board() {
  const [squares, setSquares] = useState(Array(9).fill(null));

  return (
    <>
      <div className="board-row">
        <Square value={squares[0]} onSquareClick={handleClick} />
        //...
  );
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Por último, define la función handleClick dentro del componente Board para actualizar la matriz squares que contiene el estado de tu tablero La función handleClick crea una copia de la matriz squares (nextSquares) con el método JavaScript slice() Array. Luego, handleClick actualiza la matriz nextSquares para agregar X al primer cuadrado (índice [0]).

Llamar a la función setSquares le permite a React saber que el estado del componente ha cambiado. Esto activará una nueva representación de los componentes que usan el estado de squares (Boards), así como sus componentes secundarios (los componentes squares que forman el tablero).

Figura 49:
Función HandleClick asignando el valor X

```
export default function Board() {
  const [squares, setSquares] = useState(Array(9).fill(null));

  function handleClick() {
    const nextSquares = squares.slice();
    nextSquares[0] = "X";
    setSquares(nextSquares);
  }

  return (
    // ...
  )
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Cuando estabas pasando onSquareClick={handleClick}, estabas pasando la función handleClick como una prop. No lo estábamos llamando. Pero ahora la estamos llamando a esa función de inmediato, observa los paréntesis en handleClick(0), y es por eso que se ejecuta demasiado rápido.

Podrías arreglar esto creando una función como handleFirstSquareClick que llame a handleClick(0), una función como handleSecondSquareClick que llame a handleClick(1), y así sucesivamente. Es decir, podríamos pasar (en lugar de llamar) a estas funciones como props de la forma onSquareClick={handleFirstSquareClick}. Esto resolvería el bucle infinito.

Por ejemplo, observemos la nueva sintaxis () =>. Aquí, () => handleClick(0) es una función de flecha, que es una forma más corta de definir funciones. Cuando se hace clic en el cuadrado, se ejecutará el código después de la «flecha» =>, llamando a handleClick(0).

Figura 50:
Función Flecha para afectar a Board

```
export default function Board() {
  // ...
  return (
    <>
      <div className="board-row">
        <Square value={squares[0]} onSquareClick={() => handleClick(0)} />
        // ...
  );
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Ahora necesitas actualizar los otros ocho cuadrados para llamar a handleClick desde las funciones de flecha que pasas dado que la opción anterior se presentaba para una sola aplicación al ser este una matriz debemos iterar por cada posición es decir que van a existir 9 posiciones a las cuales demos aplicar la función Square.

A estas se les debe asignar una posición donde se va a almacenar su valor por ende debemos generarlo de manera ordenada, así como se muestra en la figura 51.

Figura 51:
Función Board con subfunciones Flecha para almacenar el valor.

```
export default function Board() {
  // ...
  return (
    <>
      <div className="board-row">
        <Square value={squares[0]} onSquareClick={() => handleClick(0)} />
        <Square value={squares[1]} onSquareClick={() => handleClick(1)} />
        <Square value={squares[2]} onSquareClick={() => handleClick(2)} />
      </div>
      <div className="board-row">
        <Square value={squares[3]} onSquareClick={() => handleClick(3)} />
        <Square value={squares[4]} onSquareClick={() => handleClick(4)} />
        <Square value={squares[5]} onSquareClick={() => handleClick(5)} />
      </div>
      <div className="board-row">
        <Square value={squares[6]} onSquareClick={() => handleClick(6)} />
        <Square value={squares[7]} onSquareClick={() => handleClick(7)} />
        <Square value={squares[8]} onSquareClick={() => handleClick(8)} />
      </div>
    </>
  );
};
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

Ahora es el momento de corregir un defecto importante en este juego de tres en línea: las «O» no se pueden marcar en el tablero. Establece que el primer movimiento sea «X» de forma predeterminada. Hagamos un seguimiento de esto agregando otra parte del estado al componente Board

Cada vez que un jugador se mueve, xIsNext (un valor booleano) se invertirá para determinar qué jugador es el siguiente y se guardará el estado del juego. Actualiza la función handleClick de Board para cambiar el valor de xIsNext

Figura 52:
Asignar función IsNext para determinar nuevo jugador

```
export default function Board() {
  const [xIsNext, setXIsNext] = useState(true);
  const [squares, setSquares] = useState(Array(9).fill(null));

  function handleClick(i) {
    const nextSquares = squares.slice();
    if (xIsNext) {
      nextSquares[i] = "X";
    } else {
      nextSquares[i] = "O";
    }
    setSquares(nextSquares);
    setXIsNext(!xIsNext);
  }

  return (
    //...
  );
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

¡La X se sobrescribe con una O! Si bien esto agregaría un giro muy interesante al juego, por ahora nos apegaremos a las reglas originales.

Cuando marcas un cuadrado con una X o una O, no estás comprobando primero si el cuadrado ya tiene un valor X u O. Puedes arreglar esto regresando rápidamente en el estado. Verifica si el cuadrado ya tiene una X o una O. Si el cuadrado ya está lleno, genera un return en la función handleClick, antes de que intente actualizar el estado del tablero.

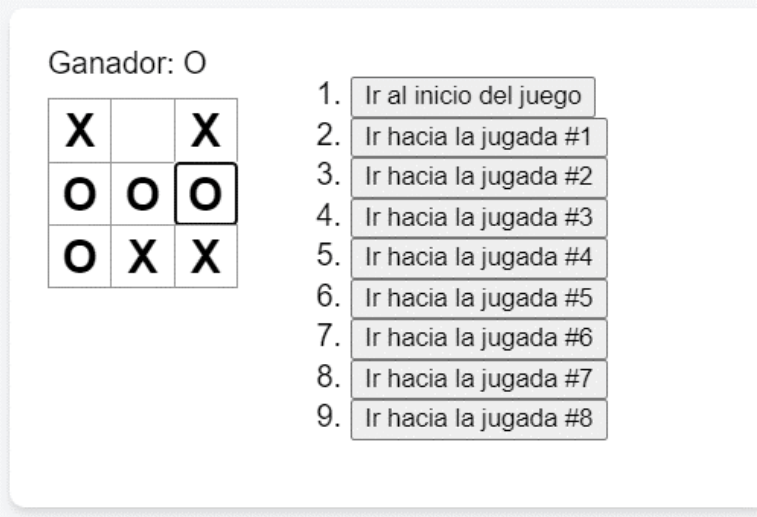
Declarar un ganador Ahora que muestras el turno del siguiente jugador, también debes mostrar cuándo se gana el juego y cuando no hay más turnos. Para hacer esto, agrega una función de ayuda llamada calculateWinner que toma una matriz de 9 cuadrados, busca un ganador y devuelve 'X', 'O' o null según corresponda. Esta función es matemática en base a los casilleros que han sido marcados predice al ganador.

Figura 53:
Función para predecir al Ganador.

```
export default function Board() {
  //...
}

function calculateWinner(squares) {
  const lines = [
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8],
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
    [0, 4, 8],
    [2, 4, 6]
  ];
  for (let i = 0; i < lines.length; i++) {
    const [a, b, c] = lines[i];
    if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {
      return squares[a];
    }
  }
  return null;
}
```

Fuente: https://es.react.dev/learn/tutorial-tic-tac-toe

A continuación, presentamos todas las funciones unificadas y que deben ir en nuestro archivo App.jsx para que se pueda ejecutar el juego de tres en raya, a su vez mencionar que deben ejecutar el comando: "npm run dev" para que este se despliegue en el navegador y puedan utilizarlo, el resultado debería visualizarse en el navegador de la siguiente manera:



4 Resolución de juego de mesa Tres en raya.

4.1 App.jsx:

```
import { useState } from 'react';
function Square({ value, onSquareClick }) {
  return (
    <button className="square" onClick={onSquareClick}>
      {value}
    </button>
  );
}

function Board({ xIsNext, squares, onPlay }) {
  function handleClick(i) {
    if (calculateWinner(squares) || squares[i]) {
      return;
    }
    const nextSquares = squares.slice();
    if (xIsNext) {
      nextSquares[i] = 'X';
    } else {
      nextSquares[i] = 'O';
    }
    onPlay(nextSquares);
  }

  const winner = calculateWinner(squares);
  let status;
  if (winner) {
    status = 'Ganador: ' + winner;
  } else {
    status = 'Siguiente jugador: ' + (xIsNext ? 'X' : 'O');
  }

  return (
    <>
      <div className="status">{status}</div>
```

```
            <div className="board-row">
                    <Square   value={squares[0]}   onSquareClick={()   =>
handleClick(0)} />
                    <Square   value={squares[1]}   onSquareClick={()   =>
handleClick(1)} />
                    <Square   value={squares[2]}   onSquareClick={()   =>
handleClick(2)} />
            </div>
            <div className="board-row">
                    <Square   value={squares[3]}   onSquareClick={()   =>
handleClick(3)} />
                    <Square   value={squares[4]}   onSquareClick={()   =>
handleClick(4)} />
                    <Square   value={squares[5]}   onSquareClick={()   =>
handleClick(5)} />
            </div>
            <div className="board-row">
                    <Square   value={squares[6]}   onSquareClick={()   =>
handleClick(6)} />
                    <Square   value={squares[7]}   onSquareClick={()   =>
handleClick(7)} />
                    <Square   value={squares[8]}   onSquareClick={()   =>
handleClick(8)} />
            </div>
          </>
        );
      }

      export default function Game() {
        const [history, setHistory] = useState([Array(9).fill(null)]);
        const [currentMove, setCurrentMove] = useState(0);
        const xIsNext = currentMove % 2 === 0;
        const currentSquares = history[currentMove];

        function handlePlay(nextSquares) {
           const nextHistory = [...history.slice(0, currentMove + 1),
nextSquares];
          setHistory(nextHistory);
          setCurrentMove(nextHistory.length - 1);
        }
```

```
        function jumpTo(nextMove) {
          setCurrentMove(nextMove);
        }

        const moves = history.map((squares, move) => {
          let description;
          if (move > 0) {
            description = 'Ir al movimiento #' + move;
          } else {
            description = 'Ir al inicio del juego';
          }
          return (
            <li key={move}>
                                        <button       onClick={()       =>
jumpTo(move)}>{description}</button>
            </li>
          );
        });

        return (
          <div className="game">
            <div className="game-board">
                   <Board  xIsNext={xIsNext}  squares={currentSquares}
onPlay={handlePlay} />
            </div>
            <div className="game-info">
              <ol>{moves}</ol>
            </div>
          </div>
        );
      }
```

```
function calculateWinner(squares) {
  const lines = [
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8],
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
    [0, 4, 8],
    [2, 4, 6],
  ];
  for (let i = 0; i < lines.length; i++) {
    const [a, b, c] = lines[i];
    if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {
      return squares[a];
    }
  }
  return null;
}
```

4.2 Index.css:

```
* {
 box-sizing: border-box;
}

body {
 font-family: sans-serif;
 margin: 20px;
 padding: 0;
}

h1 {
 margin-top: 0;
 font-size: 22px;
}
```

```
h2 {
  margin-top: 0;
  font-size: 20px;
}

h3 {
  margin-top: 0;
  font-size: 18px;
}

h4 {
  margin-top: 0;
  font-size: 16px;
}

h5 {
  margin-top: 0;
  font-size: 14px;
}

h6 {
  margin-top: 0;
  font-size: 12px;
}

code {
  font-size: 1.2em;
}

ul {
  padding-inline-start: 20px;
}

* {
  box-sizing: border-box;
}
```

```
body {
  font-family: sans-serif;
  margin: 20px;
  padding: 0;
}

.square {
  background: #fff;
  border: 1px solid #999;
  float: left;
  font-size: 24px;
  font-weight: bold;
  line-height: 34px;
  height: 34px;
  margin-right: -1px;
  margin-top: -1px;
  padding: 0;
  text-align: center;
  width: 34px;
}

.board-row:after {
  clear: both;
  content: '';
  display: table;
}

.status {
  margin-bottom: 10px;
}
.game {
  display: flex;
  flex-direction: row;
}

.game-info {
  margin-left: 20px;
}
```

Resumen del Capítulo 3

En el Tercer y último capítulo de este libro hemos visto la importancia del desarrollo de aplicaciones en la librería/framework React , lo cual nos permitió realizar un trabajo en equipo resolviendo el problema del juego de mesa tres en raya para ello se instalaron las dependencias en nuestro ambiente de trabajo, se presenta paso a paso la creación de un proyecto React-Vite y posterior a este la creación paso a paso de nuestro juego tres en raya desde la construcción del tablero hasta la resolución y creación de las marcas que tiene el usuario vencedor. Este se basó en los siguientes componentes:

- ✓ Tablero: Componente principal que contiene las celdas del juego.
- ✓ Celda: Componente individual que representa cada casilla del tablero.
- ✓ Mensaje de victoria: Componente que muestra un mensaje cuando un jugador gana.
- ✓ Tablero: Un array de arrays que representa el estado de cada celda (vacía, 'X' o 'O').
- ✓ Jugador actual: Una variable que indica si es el turno del jugador 'X' o 'O'.
- ✓ Ganador: Una variable que almacena el ganador del juego o 'empate'.

Una vez definido sus formas de actuar hemos buscado la lógica del juego de mesa y la hemos pasado a los siguientes pasos lógicos que me permiten diseñar mi juego.

- ✓ Manejo de clics: Al hacer clic en una celda, se actualiza el estado del tablero con el símbolo del jugador actual.
- ✓ Verificación de ganador: Después de cada movimiento, se verifica si hay un ganador en filas, columnas o diagonales.
- ✓ Cambio de turno: Si no hay ganador, se cambia el jugador actual.
- ✓ Reinicio del juego: Se implementa una función para reiniciar el tablero y volver a empezar.

Esto se puede implementar en un proyecto de React de la siguiente manera: Se utiliza un JSX para crear la estructura del tablero y las celdas. Luego se utiliza el estado de los componentes para almacenar la información del juego y actualizar la interfaz de usuario.

- ✓ Props: Se utilizan props para pasar información entre componentes, como el índice de una celda o el símbolo del jugador.
- ✓ Eventos: Se utilizan eventos como onClick para manejar las interacciones del usuario.
- ✓ Condicionales: Se utilizan condicionales para mostrar diferentes elementos de la interfaz de usuario según el estado del juego.

Referencias

Azaustre, C. (2023). *Aprendiendo React: Guía práctica para aprender desde cero.* Madrid: Independently published . doi:979-8852737427

Chan, M. (2016). *Building React Applications.* Estados Unidos, California: Manning Publications.

Chan, M. (2018). *Fullstack React.* Estados Unidos: Manning Publications.

Degni, R. (28 de 07 de 2023). *CodeMotion - Una completa introducción a la librería React*. Obtenido de CodeMotion - Una completa introducción a la librería React: https://www.codemotion.com/magazine/es/lenguajes-de-programacion/una-completa-introduccion-a-la-libreria-react/#:~:text=React%20fue%20creada%20por%20un,complejidad%20y%20mejorando%20el%20rendimiento.

Drasner, S. (2022). *React for Beginners: A Hands-On Guide.* Portland, USA: Frontend Masters.

Durán, M. Á. (12 de 10 de 2022). *Midudev - Es React una biblioteca o un framework*. Obtenido de Midudev - Es React una biblioteca o un framework: https://midu.dev/react-biblioteca-o-framework/

Eisenman, B. (2016). *React Quickly.* Estados Unidos, California: O'Reilly Media.

Eisenman, B. (2016). *React Quickly.* Estados Unidos, California: O'Reilly Media.

Facebook. (20 de 08 de 2013). *React Dev*. Obtenido de React Dev: https://es.react.dev/learn/start-a-new-react-project

Facebook. (2017). *React: A JavaScript library for building user interfaces.* Obtenido de https://reactjs.org/

Ferguson, N. (01 de 01 de 2023). *Medium - Single Page Applications (SPA)*. Obtenido de Medium - Single Page Applications (SPA): https://medium.com/@teamtechsis/single-page-applications-spa-48b1b845b446

Florence, R. (2023). *React Server Components.* San Francisco, USA: Sebastian Markbåge Vercel.

Khourshid, D. (2022). *React Hooks Deep Dive.* USA, California: Manning Publications.

Lonsdorf, B. (2023). *React Suspense and Concurrent Mode.* Portland, USA: Frontend Masters.

Mardan, A. (2023). *Cracking React: The Complete Guide.* Birmingham, UK: Packt Publishing.

Osmani, A. (2017). *React Design Patterns .* Estados Unidos, California: O'Reilly Media.

Quirk, J. (2021). *React and Firebase .* Birmingham, UK: Packt Publishing.

React Dev. (28 de 08 de 2024). *React Docs*. Obtenido de React Docs: https://es.react.dev/learn/tutorial-tic-tac-toe

Spinola Federico, M. (03 de 03 de 2023). *AluraCursos - Qué es Single Page Application*. Obtenido de AluraCursos - Qué es Single Page Application: https://www.aluracursos.com/blog/single-page-application

Wieruch, R. (2018). *Pro React.* Texas,Estados Unidos: Manning Publications.