

SISTEMA TODO

SISTEMAS DISTRIBUIDOS





Contenido

[Introducción](#_heading=h.gjdgxs) **2**

[Objetivo](#_heading=h.30j0zll) **2**

[Tecnologías](#_heading=h.1fob9te) **3**

[Desarrollo](#_heading=h.3znysh7) **4**

[Código del servidor](#_heading=h.2et92p0) 4

[Código del cliente](#_heading=h.tyjcwt) 4

[Conclusión](#_heading=h.3dy6vkm) **4**

[**Referencias**](#_heading=h.a28wgtkt8vit) **5**

# Introducción

Dentro del presente documento se expone la creación del sistema ToDo.

Se explicarán las tecnologías implementadas, así como el desarrollo del proyecto, y finalizamos con una breve conclusión respecto al trabajo realizado y la correlación que se tiene con la materia: sistemas distribuidos.

# Objetivo

Crear un sistema de tareas mediante websockets en la que permita a los clientes observar la lista de tareas, modificar o crear una nueva, se realizan estas acciones sin necesidad de refrescar la página.

# Tecnologías

Algunas de las tecnologías implementadas para el desarrollo del proyecto fueron:

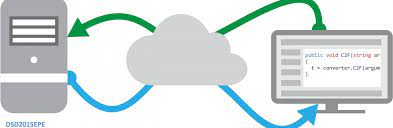
* **Node js:** Entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Node fue usado para montar nuestro servidor, ya que este fue programado mediante JavaScript y, de antemano, brinda la facilidad de implementar herramientas adicionales según se requiera.

* **React:** Librería de JavaScript de código abierto, permite la creación de interfaces para el usuario de forma rápida y sencilla, además permite adicionar tecnologías extras para usarlas según el programador de front-end lo requiera.
* **Socket.io:** Es una librería que permite implementar websockets.

Los websockets son la clave del proyecto debido a que el propósito va en implementar los sockets. Los websockets se basan en el protocolo HTTP tienen cierto parecido a los sockets.

Los sockets es la tecnología que permite la comunicación activa entre cliente y servidor, se puede decir que esta comunicación es bidireccional y lo que nos lleva a decir de cierto modo que sucede en tiempo real, puesto que el servidor no necesita recibir peticiones para brindarle al usuario la respuesta, simplemente muestra lo que tiene.



# Desarrollo

## Código del servidor



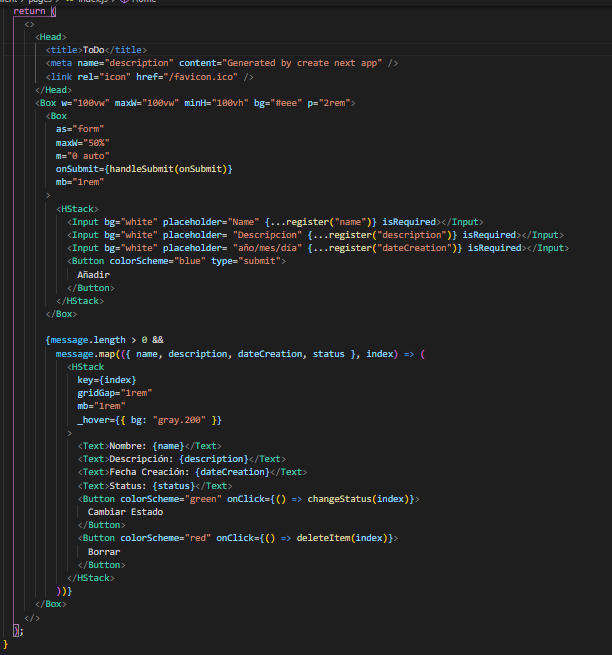
Generamos un nuevo proyecto en el que creamos nuestro servidor y pedimos que establezca la conexión mediante sockets. Al abrir la conexión el socket va a estar respondiendo mediante funciones las cuales contendrán la acción que se deben realizar en el arreglo declarado.

## Código del cliente

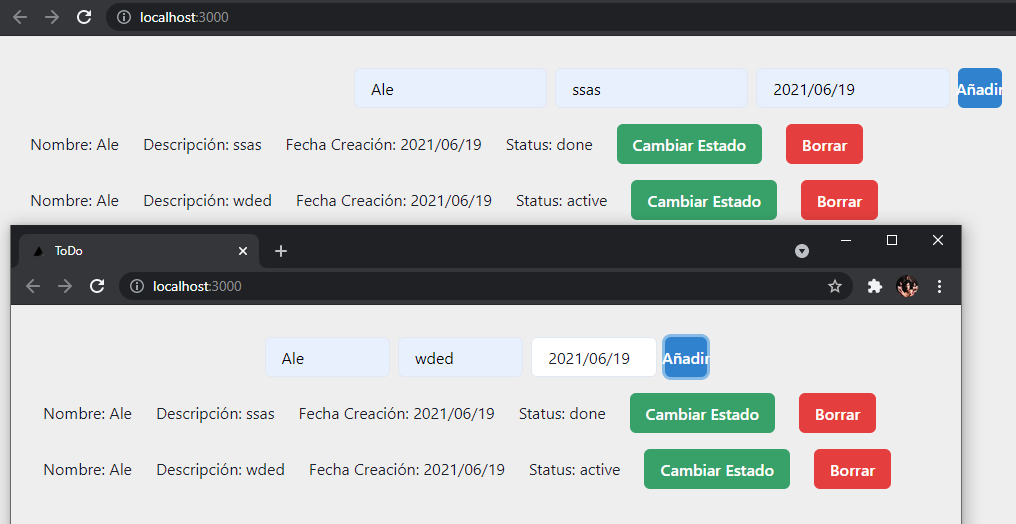


Dentro del cliente agregamos las funciones que van a interactuar con el servidor y además el html que retornará en nuestra vista.

Mediante React se vuelve más interactivo tener el html combinado con JavaScript; con apoyo de Chakra le agregamos los estilos deseados.



Como producto final tenemos un pequeño formulario para agregar tareas en el cual al ser agregada tenemos la opción de cambiar el estatus o de si queremos eliminarla, estos cambios son reflejados en todos los clientes conectados.



# Conclusión

Las similitudes de un socket a un websocket se basan en el uso, pero diferencian en implementación. Los websockets se basan en el protocolo HTTP por lo que lleva a darse cuenta que la implementación de estos es en aplicaciones web que requieren una conexión permanente al servidor. Mientras que los sockets en sí, se ejecutan a través de TCP/IP no están restringidos a navegadores ni protocolo HTTP, simplemente los sockets se diseñaron para ser independientes de la tecnología de red subyacente.

Algunos autores hacen comparativas de AJAX y los websockets, el semestre antepasado tuve la oportunidad de trabajar con AJAX el cual es basado en comunicación asíncrona (la cual permitía actualizar una sección de la página y no toda), y pude observar que, aunque no había necesidad de refrescar o actualizar la página completa, se debían de realizar peticiones mediante un único canal el cual se abría o cerraba, lo cual los websockets los lleva a hacer una notoria diferencia, pues mantienen la comunicación siempre abierta y es doble vía. Ninguna tecnología sustituye a otra, más bien todo depende de la forma en que se necesite implementar, si es necesario mantener una comunicación siempre abierta usemos websockets, de lo contrario otras tecnologías.

Con respecto a sistemas distribuidos, implementar websockets es genial, si es necesario realizar streaming o, por ejemplo, al momento de usar GPS, muchos clientes necesitamos acceder a ese recurso y no necesitemos estar actualizando la página, esto mejora la UX. Puesto que todo lo tendrán a la “mano” y en ese mismo instante. :))

# Referencias

Azaustre, C. (2015, September 24). *WebSockets: Cómo utilizar Socket.io en tus aplicaciones web*. Carlosazaustre.es. https://bit.ly/3cVrTFF

Meenakshi, A. (2021, January 8). *WebSockets tutorial: How to go real-time with Node and React*. LogRocket Blog. https://bit.ly/2UafNC2

Díaz, J. (n.d.). *Ajax y Websockets guía completa*. EDteam - No Te Detengas. Retrieved June 19, 2021, from https://bit.ly/3vLNW8h