Evaluación Next Js - Web Sockets

Tema 1

- La entrega se realiza a través de un archivo RAR con el formato de nombre. 5A_REACT_APELLIDO_NOMBRE.rar o 5B_REACT_APELLIDO_NOMBRE.rar Por ejemplo: 5X_REACT_MARCHESI_MATIAS.rar
- Si bien se pueden consultar páginas web y código previamente realizado para la resolución del examen, **cualquier intento de comunicación con un compañero** (vía Classroom, WhatsApp Web, Mail, Replit, Discord, celular, personalmente, etc.) **será motivo de anulación del examen**. A su vez, está sumamente prohibido el uso de inteligencias artificiales (chatgpt, etc)
- Recuerde apoyarse en el uso de la consola para la detección de errores.
- La web entregada puede ser lo más fea posible. No pierda tiempo intentando embellecerla. Eso sí, debe poder ser entendible y cumplir con los requisitos que se le solicitan.
- ***Comentar el código, indicar qué se esperaba hacer, qué se hizo y qué diferencia hay entre lo que se esperaba y lo que se logró hacer (esto último en caso de no haber finalizado).***
- ***En caso de finalizar el examen en tiempo y forma, comuníqueselo a los docentes previo a entregar.***

Consigna:

Realizar un programa capaz de organizar Tareas entre un grupo de trabajo.

1. Realizar un componente llamado **Tarea** que consta de 3 campos: **Nombre de la tarea**, **Responsable de la Tarea** y **Estado**. Donde estado puede tomar 3 valores: **Pendiente, En Proceso** o **Realizada**

2. Realizar una página home donde la primera vez que se carga la página se llame por medio de un fetch a un pedido **GET "/tareas"**, la cual va a traer una lista con todas las tareas en el Backend. Una vez que se tenga la lista en el Front mostrar en la página web las mismas. La lista de las tareas va a tener el siguiente formato:

```
[{
        nombre: "Crear Base de Datos",
        responsable: "Marchesi",
        estado: "Pendiente"
    },
    {
        nombre: "Crear Evaluaciones",
        responsable: "Rivas",
        estado: "Realizada"
    },
    {
        nombre: "Rendi Evaluacion",
        responsable: "Alumnos",
        estado: "En proceso"
    }
]
```

3. Sumarle a la página **dos inputs y un botón** para agregar una tarea a la lista. Para lo cual, al pulsar el botón se debe enviar un fetch al pedido **POST "/crearTarea"** con los 2 datos en un objeto cuyas parámetros sean nombre y responsable. Previo se debe verificar en esta función que no exista una tarea con ese nombre, ya que el nombre es el identificador único de las tareas. La tarea por defecto en el back se crea con estado "Pendiente".
4. Al crearse, enviar un aviso por socket para que le llegue a los otros usuarios al instante así pueden actualizar su lista.

Pista: Tener en cuenta que en todos los puntos que se pida hacer un socket se debe realizar tanto el emit (socket.emit()) como la función correspondiente para la recepción (socket.on()). En este caso se tiene que actualizar la lista de tareas. En el back esto está hecho de esta forma:

```
/*
    Data debe ser:
    {
        nombre,
        responsable
    }
*/
socket.on('tareaCreada', data =>{
    console.log("Tarea creada: ", data);
    io.emit('avisoCreacion', { event: "Creacion de tarea", message: data });
})
```

5. Sumarle a la página otros elementos: un input, un select y un botón para cambiar el estado de una tarea. En el input, el usuario tendrá que escribir el nombre de la tarea a modificar, en el select elegirá el nuevo estado (puede ser: **Pendiente**, **En Proceso** o **Realizada**). Y al presionar el botón se debe enviar un aviso mediante el socket al backend y al resto de usuarios con el nuevo estado.

Recordar:

- Hacer la función recepción del socket (socket.on()) para que al momento de recibir un cambio de estado en una de ellas, la misma se modifique en la lista de tareas.
- El backend estará a la espera de un evento "tareaModificada" y emitirá un evento "avisoModificación". Armen el objeto data de forma tal que tenga el nombre de la tarea y el estado modificado