EVALUACIÓN PARCIAL: Sistema de Gestión de Tickets de Soporte

Objetivo:
Implementar una aplicación de página única (SPA) con React para gestionar tickets de soporte. El enfoque principal está en la visualización de datos desde una API, la creación de nuevos registros, y la actualización dinámica de la interfaz, utilizando la stack tecnológica moderna vista en clase.

Contexto:
Se requiere un sistema para que un pequeño equipo de soporte técnico pueda registrar y visualizar los tickets de trabajo pendientes. La aplicación debe permitir a los usuarios ver todos los tickets existentes en una tabla y agregar nuevos tickets a través de un formulario dedicado.

Tecnologías a Utilizar:

- **Backend (Provisto por la cátedra):** Node.js, Express.js, Sequelize, SQLite. Los alumnos solo necesitan ejecutarlo.
- **Frontend:** React (usando Create React App), Componentes Funcionales, Hooks (useState, useEffect), react-hook-form (para manejo de formularios), axios (para comunicación con el backend), Bootstrap 5.
- **Entorno de desarrollo:** Visual Studio Code (VS Code), Node.js.

Prerrequisitos:

- Descargar **Parcial2_DDS_3k4_19_06_2025.zip** y descomprimir en su carpeta de trabajo.
- El ZIP contendrá dos carpetas: **backend** y **frontend**.
- En una terminal, navegar a la carpeta backend y ejecutar `npm install` para instalar sus dependencias.
- En otra terminal, navegar a la carpeta frontend y ejecutar `npm install` para instalar las dependencias de React.

REQUISITOS FUNCIONALES:

1. **Listado de Tickets (Componente funcional 1):**
   Al cargar la aplicación, se debe mostrar un listado de todos los tickets de soporte registrados en la base de datos, presentados en una tabla.

2. **Creación de un Nuevo Ticket (Componente funcional 2):**
   La aplicación debe incluir un formulario con los siguientes campos:

   - Nombre de la Tarea (texto, obligatorio).
   - Fecha (fecha, obligatoria).
   - Prioridad (número entero entre 1 y 10, obligatorio).

3. **Actualización Dinámica:**
   Al enviar exitosamente el formulario de creación, el nuevo ticket debe aparecer automáticamente en el listado de tickets sin necesidad de recargar la página manualmente.

REQUISITOS TÉCNICOS:

1. **Estructura de Proyecto:**
   Se proporciona el proyecto React (frontend) como base para el desarrollo de los componentes funcionales solicitados.

2. **Backend (Provisto):**
   Se proporciona un backend funcional que expone los siguientes endpoints de la API REST:

   - `GET /api/tickets`: Retorna un array JSON con todos los tickets.
   - `POST /api/tickets`: Recibe un objeto JSON en el body para crear un nuevo ticket. El body debe tener la forma:

     ```
     { "nombreTarea": "...", "fecha": "YYYY-MM-DD", "prioridad": X }
     ```

     Retorna el ticket creado.

   - El modelo de datos de un Ticket es: `idTicket (PK)`, `nombreTarea`, `fecha`, `prioridad`.

3. **Frontend (A desarrollar por el alumno):**

   - **Componente Principal (App.js):** Debe ser el componente que orquesta la aplicación. Se recomienda que maneje el estado principal (la lista de tickets) y lo pase como props a los componentes hijos.

   - **Componente TicketList.jsx:**

     - Debe ser un componente funcional responsable de renderizar la tabla de tickets.

     - Recibirá la lista de tickets como una prop desde App.js.

     - Debe mostrar un mensaje como "Cargando tickets..." mientras se obtienen los datos.

     - Debe mostrar un mensaje como "No se encontraron tickets" si la lista está vacía.

     - Debe utilizar el método `.map()` para generar las filas (`<tr>`) de la tabla dinámicamente, asignando una key única a cada fila.

   - **Componente TicketForm.jsx:**

     - Debe ser un componente funcional que contenga el formulario de creación.

     - Debe utilizar la librería `react-hook-form` para manejar el estado del formulario, el registro de los inputs (`register`), la gestión del envío (`handleSubmit`) y las validaciones.

     - Al enviar el formulario, debe realizar una petición `POST` a `/api/tickets` con los datos del formulario.

     - Tras una creación exitosa, debe comunicarse con el componente padre (`App.js`) para indicarle que la lista de tickets debe ser actualizada. Se sugiere para esto pasar una función de "refresco" como prop desde `App.js` al formulario.

   - **Manejo de Datos y Estado:**

     - Utilizar el hook `useState` en `App.js` para almacenar el array de tickets.

- Utilizar el hook `useEffect` en `App.js` para realizar la petición `GET` inicial a `/api/tickets` y mostrar los tickets por primera vez.

- **Estilos:** Utilizar clases de Bootstrap 5 para dar un aspecto profesional y ordenado a la tabla (`table`, `table-striped`), al formulario (`form-control`, `btn`, `btn-primary`) y al layout general (`container`, `row`, `col`).

PUNTOS DE PARTIDA SUGERIDOS:

Se proporcionan esqueletos de archivos dentro de la carpeta `frontend/src`: `App.js`, `components/TicketList.jsx` y `components/TicketForm.jsx` con una estructura básica para que el alumno complete la lógica.

INSTRUCCIONES PARA LA ENTREGA Y EVALUACIÓN:

- Al finalizar el parcial, deberán **comprimir todo el contenido** de la carpeta `frontend`, **excluyendo expresamente** la carpeta `node_modules` y la carpeta `build`.

- El archivo comprimido debe ser un archivo ZIP y tener el siguiente formato: `parcial2_react_3k4_<legajo>.zip` (reemplazar `<legajo>` con su número de legajo).

- El archivo ZIP deberá ser **subido a la plataforma Moodle** dentro del tiempo estipulado.

- Deberán permanecer en su estación de trabajo con el proyecto abierto en VS Code. Para la evaluación presencial, el servidor backend (`npm start` en la carpeta backend) y el servidor de desarrollo de React (`npm start` en la carpeta frontend) deben estar funcionando y listos para ser evaluados.

CRITERIOS DE EVALUACIÓN (Rúbrica - 10 puntos en total):

1. **Entorno y Componente de Listado (TicketList.jsx):**
    - **(3 puntos)** Implementación correcta del hook `useEffect` en el componente `App.js` para hacer el fetch inicial de datos (`GET /api/tickets`).
        i. Uso correcto de `useState` para almacenar los tickets.
        ii. Pasaje de los datos vía props al componente `TicketList` y renderizado correcto de la tabla.
        iii. Manejo de estados de "Cargando..." y "No hay tickets".
        iv. Fundamentación oral.
2. **Componente de Formulario (TicketForm.jsx):**
    - **(3 puntos)** Implementación correcta de `react-hook-form`.
        i. Uso del hook `useForm` para obtener `register`, `handleSubmit` y `formState: { errors }`.
        ii. Asociación de los inputs del formulario usando `register` y aplicación de validaciones básicas (ej. `required`).
        iii. Fundamentación oral.
3. **Creación de Tickets (Lógica de API y Envío):**
    - **(2 puntos)** Implementación correcta de la función `onSubmit` que se pasa a `handleSubmit`.
        i. Realización de la petición `POST` al backend con los datos del formulario.
        ii. Manejo de la respuesta de la API.
        iii. Fundamentación oral.

4. **Manejo de Estado y Comunicación entre Componentes:**
    - **(1.5 puntos)** Implementación de la actualización automática de la lista de tickets después de una creación exitosa.
        i. Comunicación del `TicketForm` al `App.js` (por ejemplo, vía una función callback pasada por props).
        ii. Fundamentación oral.
5. **Estructura y Buenas Prácticas de React y HTML:**
    - **(0.5 puntos)** Organización del código en componentes funcionales claros.
        i. Uso correcto de `key` en los listados.
        ii. Aplicación correcta de clases de Bootstrap para una interfaz limpia y funcional.
        iii. Fundamentación oral.

CONDICIÓN DE APROBACIÓN:

Para aprobar la evaluación, se debe alcanzar un mínimo de **6 puntos (60%)**. Esto requiere que las funcionalidades de **listar y crear tickets estén implementadas de extremo a extremo** (backend funcionando, frontend consume `GET`, renderiza, consume `POST`, y actualiza la UI dinámicamente), y que el alumno pueda responder satisfactoriamente las preguntas de coloquio que se realicen para fundamentar su código.