Prueba de desempeño – Módulo 3

JavaScript

Caso de uso: *eres un desarrollador web que ha recibido el encargo de desarrollar una Single Page Application (SPA) dedicada a la gestión de eventos, diseñada para que un organizador de eventos pueda gestionar una serie de eventos basados en la disponibilidad de lugares y asistentes. Este proyecto incluye la implementación de funcionalidades clave como la* ***autenticación de usuarios****,* ***gestión de rutas protegidas****, y* ***persistencia de sesión****, utilizando tecnologías modernas de JavaScript, HTML5, y CSS.*

Sumado a lo anterior, deberás ***simular una base de datos*** *utilizando* ***json-server*** *para realizar operaciones CRUD, asegurando la consistencia y la integridad de los datos. Esta SPA proporcionará una experiencia completa tanto para los administradores, que podrán gestionar eventos y asistentes, como para los visitantes, que podrán ver y registrarse en los eventos disponibles.*

Funcionalidades principales: para alcanzar un resultado óptimo en esta prueba, deberás cumplir cada uno de los siguientes requisitos:

Diseño guía: clic aquí para ir al diseño guía *(Este diseño se basó en el trabajo de Apple. K (2021), rediseñado por Domínguez. A (2025))*

Requisitos:

1. **Sistema de autenticación:**
    a. Registro de usuarios con dos roles: administrador y visitante.
    b. Inicio de sesión para usuarios registrados.
    c. Protección de rutas mediante un guardián en Router.js.
2. **Persistencia de sesión:**
    a. Uso del Local Storage para mantener la sesión iniciada y garantizar la experiencia del usuario.
    b. Al iniciar sesión, la información del usuario debe almacenarse en Local Storage para mantener la sesión.
    c. La sesión debe persistir incluso al recargar la página.
3. **Consistencia de datos:**
    a. La aplicación debe sincronizar correctamente las operaciones CRUD con la base de datos simulada (json-server).
4. **Interfaz de usuario:**
    a. La SPA debe ser responsiva y proporcionar una experiencia de usuario fluida.
    b. Debe haber formularios intuitivos para el registro y login, así como para la gestión de eventos.

Criterios de aceptación:

1. **Funcionalidad completa:**
    a. Los usuarios pueden registrarse, iniciar sesión, y navegar por las rutas según su rol asignado.

b. Los administradores pueden realizar todas las operaciones CRUD relacionadas con los eventos.
c. Los visitantes pueden visualizar y registrarse en los eventos disponibles.

2. **Persistencia de sesión:**
   a. La sesión del usuario debe mantenerse activa entre recargas de página.
3. **Consistencia de datos:**
   a. Los datos deben sincronizarse correctamente entre la aplicación y json-server.
4. **Interfaz de usuario:**
   a. La SPA debe ser responsiva y permitir una navegación fluida entre sus vistas.
5. **Entrega y documentación:**
   a. Los archivos están organizados y presentes en el repositorio del proyecto.
   b. El código contiene comentarios claros explicando las secciones clave.
   c. El repositorio evidencia commits descriptivos por funcionalidad.
6. **Tipos de usuarios:**
   a. **Usuario administrador:**
      i. En el archivo db.json, debe existir un usuario con el rol de administrador por defecto. Este usuario podrá iniciar sesión con este rol.
      ii. El usuario administrador tendrá las capacidades de editar y eliminar eventos.
   b. **Usuario visitante:**
      i. Los visitantes podrán registrarse en eventos siempre y cuando no se haya superado la capacidad del evento.
      ii. En la misma vista, los visitantes podrán visualizar sus registros.
7. **Lógica de rutas:**
   a. Si el usuario no está autenticado e intenta acceder a una de las siguientes rutas, deberá ser redirigido a una página custom not-found.js.
   b. Si el usuario ya se encuentra autenticado e intenta acceder a la ruta /login o /register, deberá ser redirigido a la ruta raíz del dashboard /dashboard.
8. **Vistas:**
   a. Vista de Home - Path: /dashboard
   b. Vista de Crear Eventos - Path: /dashboard/events/create
   c. Vista de Editar Eventos - Path: /dashboard/events/edit

Entregables:

1. Enlace al repositorio en GitHub (público).
2. El repositorio debe contener un archivo README con instrucciones detalladas sobre el proyecto, además de la información del coder (Nombre, Clan, correo, documento de identidad). Este archivo debe detallar paso a paso cómo levantar y usar la solución, garantizando que el Team Leader no tenga que realizar ingeniería inversa o adivinar cómo el proyecto se corre.
3. Archivo db.json configurado con json-server.
4. Colección POSTMAN que permita probar la solución.

Consideraciones generales:

- Deberás generar un proyecto de Node con el comando visto en clase.

- El nombre del proyecto en el archivo package.json debe ser tu nombre completo en minúscula sin espacios y los últimos 3 dígitos de tu cédula. Ej: nombreapellido213
- La aplicación debe ser funcional. Enfócate en la lógica con JavaScript antes que en el aspecto de la página.

Recursos:

- Documentación oficial de JS
- Los siguientes archivos te serán proporcionados mediante moodle.

```
├ app/              # Carpeta de código fuente
├ .gitignore        # Archivo de ocultar archivos/directorios a Git
├ index.html        # Archivo principal de HTML
├ index.js          # Archivo principal de javaScript
├ package.json      # Dependecias del proyecto y scripts
└ README.md         # Archivo readme con instrucciones
```