Prueba de desempeño del módulo JavaScript

Clan: Hopper

Reglas de la prueba

- **Comunicación:** No está permitido hablar o comunicarse de cualquier forma con otros estudiantes durante la prueba.
- **Integridad Académica:** Cualquier forma de trampa, incluido el plagio, copia o uso de material no autorizado (autogeneradores de código o inteligencia artificial), resultará en una calificación de cero en el examen y puede llevar a sanciones adicionales según las políticas de RIWI.
- **Material Permitido:** Solo se permite ver material de apoyo como lo son diapositivas o ejercicios realizados en clase, y notas tomadas en el cuaderno. Están permitidas las siguientes páginas para acceso:
    - https://www.w3schools.com/
    - https://developer.mozilla.org/es/
    - https://boxicons.com/
    - *Cualquier otra página que no se encuentre en una de las anteriores, deberá ser informada al team lead.*
- **Permanencia en el Aula:** Una vez iniciado el examen, no se permite salir del aula hasta haber entregado el examen y descanso cada 3 horas.
- **Entrega:** Una vez finalizada la prueba se debe subir en una carpeta comprimida, con el siguiente nombre: Prueba3_NombreApellido del coder al Moodle, en la carpeta comprimida deben estar el ejercicio, la carpeta con el contenido. La hora máxima de entrega es a las (10:00 pm) ***las pruebas enviadas después de esta hora no se tendrán en cuenta.***

Se desea crear una Single Page Application (SPA) de reserva de vuelos, en la cual un agente de vuelo tendrá la capacidad de gestionar una serie de reservas basadas en los vuelos disponibles según la flota existente.

Esta prueba está diseñada para evaluar tus habilidades en desarrollo web mediante la creación de una aplicación completa que incorpora varias funcionalidades clave. Se espera que construyas un sistema de autenticación que permita el registro y login de usuarios, así como la protección de rutas sensibles mediante un guardián. Además, deberás implementar la persistencia de sesión utilizando Localstorage para asegurar que los usuarios permanezcan autenticados entre sesiones del navegador.

En resumen la aplicación web debe contar con las siguientes funcionalidades generales:

1. **Sistema de Autenticación:**
    - Registro de usuarios (rol administrador, rol visitante).
    - Login de usuarios.
    - Protección de rutas mediante un guardián en Router.js
2. **Persistencia de Sesión:**
    - Uso del Local Storage para mantener la sesión iniciada.
3. **Consistencia de Datos:**
    - Uso de **json-server** para simular una base de datos y manejar operaciones CRUD.
4. **Encarpetado de Proyecto**
    - Generar una organización a nivel estructural de los archivos, para generar una separación de responsabilidades y que sea más sencillo navegar en el proyecto. Genere los archivos que considere necesarios.

      Ejemplo:

      Carpetas de **assets**, **componentes**, **utilidades**, **servicios**, entre otros.

5. **Sistema de Administración**
    - Apartado para usuario administrador que permita la gestión completa de una entidad. (Crear, Actualizar, Eliminar, Obtener).
6. **Vista General Usuario Visitante**
    - Los usuarios visitantes tendrán una vista en la que solo se les permitirá ver los elementos disponibles. Se les debe bloquear el acceso a las acciones de Crear, Editar y Eliminar.

Descripción del problema

Se desea crear una Single Page Application (SPA) dedicada a la reserva de vuelos, diseñada para que un agente de vuelo pueda gestionar una serie de reservas basadas en los vuelos disponibles y la flota existente. Este proyecto incluye la implementación de funcionalidades clave como la autenticación de usuarios, gestión de rutas protegidas, y persistencia de sesión, utilizando tecnologías modernas JS, HTML5, JavaScript. Además, se deberá simular una base de datos utilizando json-server para realizar operaciones CRUD, asegurando la consistencia y la integridad de los datos. Está SPA proporcionará una experiencia completa tanto para los administradores, que podrán gestionar vuelos y reservas, como para los visitantes, que podrán realizar reservas de vuelos disponibles.

Para esto, debes asegurarte de lo siguiente:

Tipos de usuarios

1. **Usuario administrador**
    a. En el archivo db.json, debe existir un usuario con el rol de administrador por defecto. Este usuario podrá iniciar sesión con este rol.
    b. El usuario administrador tendrá las capacidades de editar y eliminar reservas.
2. **Usuario visitante**
    a. Los visitantes podrán reservar vuelos siempre y cuando no se haya superado la capacidad del vuelo.
    b. En la misma vista, los visitantes podrán visualizar sus reservas.

Lógica de Rutas:

1. Si el usuario no está autenticado e intenta a acceder a una de la siguientes rutas, deberá ser redirigido a una pagina custom ***not-found.js***
2. 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.

Persistencia

3. **Estructura del Archivo db.json:**
    a. El archivo db.json contendrá las siguientes entidades:
        i. Vuelos: Con información preestablecida bajo la siguiente interfaz:

```
interface Flight {
  id: number; // Flight ID
  number: number; // Flight number
  origin: string; // Origin airport code it could be: JFK, LAX, MIA or CDG
  destination: string; // Destination airport code it could be: JFK, LAX, MIA or CDG
  departure: Date; // Departure date and time
  arrival: Date; // Arrival date and time
  capacity: number; // Total number of seats available
}
```

        ii. Roles: Con dos valores fijos: Usuario Visitante y Usuario administrador. Debe cumplicar la siguiente interfaz:

```
interface Role {
  id: number; // Role ID
  name: string; // Role name
}
```

        iii. Usuarios: Inicialmente vacío, excepto por el usuario el cual debe estar creado previamente el usuario administrador. Debe cumplir la siguiente interfaz

```ts
interface User {
  id: number; // User ID
  name: string; // User name
  email: string; // User email
  birthdate: Date; // User birthdate
  password: string; // User password
  roleId: number; // Role ID
}
```

iv. Reservas: Inicialmente vacío. Debe cumplir la siguiente interfaz

```ts
interface Booking {
  id: number; // Booking ID
  flightId: number; // Flight number
  userId: number; // Customer ID
  bookingDate: Date; // Booking date
}
```

4. **LocalStorage**

Vistas

5. **Vista de Registro - Path: /register**
   a. En la vista de registro, se le pedirá al usuario que proporcione la siguiente información:
      i. Nombre
      ii. Correo electrónico
      iii. Fecha de nacimiento
      iv. Contraseña
   b. Criterios de aceptación
      i. Se debe verificar que el correo electrónico sea válido desde la etiqueta HTML y desde JavaScript, utilizando una función llamada emailValidator() la cual retorna un booleano, esta función debe evaluar que incluya un arroba @ y un punto .

6. **Vista de Login - Path: /login**
    a. En la vista de login se le pedirá al usuario la siguiente información
        i. Correo electrónico
        ii. Contraseña
    b. Criterios de aceptación
        i. Si el correo electrónico y contraseña son correctos se debe ingresar a un dashboard de layout propio el cual tendrá un navbar fijo con los siguientes items en el menú:

            Reservar

7. **Vista de Home - Path: /dashboard**
    a. La vista de HOME tendrá las siguiente opciones comunes para ambos roles:
        i. Sección superior Con título “Vuelos actuales”, el cual debe mostrar todos los vuelos, en una tarjeta cada vuelo, con información completa (la disposición de los elementos queda a tu criterio), a excepción del id.
    b. Para rol Administrador:
        i. Para cada fila de la sección común “Todos los vuelos”, solo al administrador se le habilitará dos botones: editar y eliminar.
            1. Si el usuario administrador presiona editar, lo debe redirigir a la vista de edición del vuelo en cuestión.
            2. Si el usuario administrador presiona eliminar, una alerta de tipo confirmación debe dispararse y preguntar si realmente desea eliminar el vuelo. Acto seguido, se eliminar del archivo db.json o db.json5
        ii. Tendrá, justo debajo de la vista de todos los vuelos, un botón “crear vuelo” el cual me debe redirigir a la vista de “crear vuelos”.
    c. Para rol Invitado
        i. Para cada fila de la sección común “Todos los vuelos”, solo al usuario invitado se le habilitará un botón “reservar”, en cada fila de vuelos.
            1. Si el usuario da click en el botón “reservar” un pop up de tipo ***confirm*** aparecerá y le informará al usuario si realmente quiere reservar.

8. **Vista de Crear Vuelos - Path: /dashboard/flights/create**
    a. Esta vista **solo** debe estar habilitada para el rol “administrador”
    b. Una vez dentro, el administrador verá los campos a rellenar del formulario:
        i. Flight Number: Numero de vuelo
            1. **Type**: string
            2. **Máximum**: 20 caracteres
        ii. origin: Lugar de origen
            1. **Type**: string
            2. **Máximum**: 50 caracteres
        iii. destination: Lugar de destino
            1. **Type**: string
            2. **Máximum**: 50 caracteres
        iv. departure: Fecha de salida en Formato YYYY-MM-DDT00:00:00
            1. **Type**: Date
        v. arrival
            1. **Type**: Date
9. **Vista de Editar Vuelos - Path: /dashboard/flights/edit**
    a. Esta vista **solo** debe estar habilitada para el rol “administrador”
    b. Una vez dentro, el administrador verá los campos del formulario llenos por defecto, con base en el id.
        i. Solamente se pueden editar los siguiente campos:
            1. Fecha de origen y Fecha de Llegada
            2. Capacidad

Recursos

Los siguiente archivos te serán proveídos en el siguiente link el cual se habilitará previo a la prueba de desempeño.

- .babelrc
- .gitignore
- webpack.config.js
- README.md
- api.http
- data/
    - db.json
- app/

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 ultimos 3 digitos de tu cédula:
    - Ej: nicolaspiconjaimes213
- La aplicación debe ser funcional, enfócate en la lógica con Javascript antes que el aspecto de la página.