Sistema de Gestión Hotelero - Documentación por Módulos - Usuarios

Programa:
Tecnólogo en Análisis y Desarrollo de Software
2977341

Sede:
SENA Regional Boyacá
Centro de Desarrollo Agropecuario y Agroindustrial

Evidencia:
GA8-220501096-AA1-EV02

Realizado por:
María Jimena Sánchez Pinilla
Instructor:
Jorge Eliecer Niño Ochoa

Fecha:
15 de junio de 2025

Documentación por Módulos

1. Módulo: Registro de Usuarios

- **Nombre de los Archivos**: UserModel.js, UserController.js, UserRoutes.js, SignUp.jsx, UserAdmin.jsx
- **Descripción**: Permite registrar nuevos usuarios en la base de datos del hotel.
- **Datos de entrada**:
  - "name": "texto",
  - "last_name": "texto",
  - "id_type_id": número,
  - "id_number": "texto",
  - "phone": "texto",
  - "bith_date": "fecha",
  - "email": " texto",
  - "id_user_type": número,
  - "password": " texto"
- **Proceso**:
  - Desde el frontend enviar al backend los datos suministrados por el usuario.
  - Desde el backend Validar que se hayan ingresado todos los campos.
  - Verificar que el usuario no exista a través del email.
  - Si el usuario no existe:
    - Encriptar la contraseña.
    - Crear el usuario en el base de datos.
  - Si el usuario ya existe:
    - Insertar los datos en la base de datos.
    - Notificar con un mensaje.
  - Solicitar la respuesta del backend.
  - Parsear la respuesta a JSON.
- **Datos de salida**:
  - Si no se suministran todos los datos obligatorios: mensaje de error.
  - Si el usuario ya existe: mensaje de error.

- Si el usuario es creado en la base de datos: mensaje de éxito.

2. Módulo: Inicio de Sesión

- **Nombre del Archivo**: UserModel.js, UserController.js, UserRoutes.js, SignIn.jsx
- **Descripción**: Permite ingresar al panel de clientes o administradores según el rol que se tenga.
- **Datos de entrada**:
  - "email": " texto",
  - "password": " texto"
- **Proceso**:
  - Desde el frontend enviar al backend los datos suministrados por el usuario.
  - Desde el backend validar que se hayan ingresado todos los campos.
  - Verificar que el usuario exista.
  - Si el usuario existe:
    - Comparar la contraseña.
    - Si la contraseña es igual:
      - Es autenticado el usuario y se envía la respuesta al frontend.
      - Desde el frontend se parsea la respuesta a JSON.
      - Se redirecciona el usuario al panel de clientes o al administrativo.
    - Si la contraseña no es igual:
      - No es autenticado el usuario y se envía la respuesta al frontend.
      - Desde el frontend se parsea la respuesta a JSON.
      - Se muestra un mensaje de error.
- **Datos de salida**:
  - Si no se suministran todos los campos necesarios: mensaje de error.
  - Si se suministran todos los campos, pero la contraseña es incorrecta: mensaje de error.
  - Si se suministran todos los campos, pero el correo es incorrecto: mensaje de error.
  - Si se suministran todos los campos, la contraseña y el correo son correctos: mensaje de éxito.

3. Módulo: Consultar Usuarios

- **Nombre del Archivo**: UserModel.js, UserController.js, UserRoutes.js, UserAdmin.jsx
- **Descripción**: Permite consultar todos los datos de un usuario desde el panel administrativo a través del número del documento de identificaicón.
- **Datos de entrada**: id_number
- **Proceso**:
  - Desde el frontend se presiona el botón Consultar.
  - Se llama a la función del backend y se envía el id_number del usuario.
  - En el backend se recibe el id_number y se comunica con la base de datos para seleccionar el usuario correspondiente.
  - Se verifica si el usuario existe.
  - Si el usuario no existe:
    - Se envía un mensaje de error.
  - Si el usuario existe:
    - Se envían los datos del usuario.
    - La respuesta es parseada a JSON en el frontend.
    - Se abre el modal con la información recibida.
- **Datos de salida**:
  - Si el id_number no existe: mensaje de error.
  - Si el id_number sí existe: mensaje de éxito.

4. Módulo: Modificar Usuarios

- **Nombre del Archivo**: UserModel.js, UserController.js, UserRoutes.js, UserAdmin.jsx
- **Descripción**: Permite modificar o actualizar los datos de los usuarios registrados en la base de datos.
- **Datos de entrada**: Se pueden modificar los siguientes datos:
  - "name": "texto",
  - "last_name": "texto",
  - "id_type_id": número,
  - "id_number": "texto",

- "phone": "texto",
- "birth_date": "fecha",
- "email": " texto",
- "id_user_type": número,
- "password": " texto"

- **Proceso**:
  - El usuario selecciona el botón modificar.
  - A través del ID se reciben los datos actuales del usuario.
  - Se abre el modal con la tabla y formulario para modificar la información. En la tabla se muestra los datos actuales y los nuevos valores. La solicitud envía todos los campos, aquellos que no se modifican se mantienen igual.
  - El usuario modifica los datos y presiona el botón guardar.
  - El frontend envía una solicitud al backend con los datos del formulario.
  - Los datos enviados son recibidos por el backend, se verifica si el ID fue proporcionado y si todos los campos están llenos.
  - Si no se proporciona el ID no funciona la operación.
  - Si no están llenos todos los campos del formulario envía un mensaje de error
  - Si se proporciona el ID y todos los campos del formulario, se valida uno por uno para encontrar cambios.
  - Ya que, al leer los datos del usuario se están solicitando a la base de datos campos que no existen como una columna en la tabla de “user”, el backend elimina dichos datos.
  - El backend se comunica con la base de datos y envía los datos.
  - La base de datos recorre cada uno de los campos o filas y asignan los valores.
  - El backend envía la respuesta, si es exitosa o tiene algún error retorna un mensaje.

- **Datos de salida**:
  - Si no se proporciona el ID del usuario: no funciona la operación.
  - Si se proporciona el ID y todos los campos del formulario: mensaje de éxito.
  - Si no se llenan todos los campos del formulario: mensaje de error.
  - Si se presiona el botón “Guardar”, pero no se han realizado cambios en ningún campo del formulario: mensaje de error.

5. Módulo: Eliminar Usuarios

- **Nombre del Archivo**: UserModel.js, UserController.js, UserRoutes.js, UserAdmin.jsx
- **Descripción**: Permite eliminar usuarios desde el panel administrativo.
- **Datos de entrada**: id_number
- **Proceso**:
  - Un usuario administrativo presiona el botón “Eliminar” en uno de los usuarios existentes y se abre el modal para confirmar la operación.
  - Si el usuario confirma la eliminación el frontend solicita una respuesta al backend.
  - El backend toma los datos enviados y se comunica con la base de datos.
  - La base de datos toma el id_number y ejecuta la solicitud.
  - Cuando el backend recibe la respuesta de la base de datos verifica si alguna fila fue afectada, es decir, si ese id_number existe.
  - Si ninguna fila fue afectada notifica un mensaje de error.
  - El backend también verifica si el usuario tiene reservas en curso, realizando una consulta a la base de datos con el id del usuario.
  - La base de datos toma el id y ejecuta la búsqueda.
  - Si tiene reservas en curso no permite eliminar al usuario y notifica un mensaje de error.
  - Si alguna fila en la base de datos fue afectada y el usuario no tiene reservas en curso el sistema elimina al usuario.
- **Datos de salida**:
  - Si en la base de datos no se encontró el id_number: mensaje de error.
  - Si el usuario tiene reservas en curso: mensaje de error.
  - Si fue encontrado el id_number y no tiene reservas en curso: mensaje de éxito.