Actividad 1– Continúa con tu proyecto puesto en funcionamiento con una herramienta de construcción de proyectos.

Eventos (10 puntos)

Se te proporciona:

- La API sobre MongoDB: **Eventos**, que incorpora la gestión (GET, GET por id, POST y DELETE por id) de los Eventos y la autenticación de usuarios. Para obtener, añadir y/o borrar Eventos se necesita un **token** de autenticación.
- Las páginas **login.html** y **register.html** para implementar la gestión de registro de usuarios y su autenticación en la aplicación.

Se te solicita:

1. Modifica el método/s dónde realices el envío de las peticiones al servidor de la API para añadir la comprobación de si existe el token, enviarlo en la cabecera **Authorization** antes de hacer la petición: (0,5 puntos)

   ```
   const token = localStorage.getItem("token");

   if(token) headers.Authorization = "Bearer " + token;
   ```

2. Crea la clase **Auth** en **auth-service.js** para implementar la gestión del *login del usuario*. Implementará los siguientes métodos: (3 puntos)

   – login

   Recibirá un `username` y un `password` y hará una llamada **POST** al servicio en la url **${SERVER}/auth/login**, pasándole ambos datos en un objeto JSON.

   Ejemplo: `{username: "marina ", password: "1234"}`

   Si el login es correcto, el servidor devolverá un objeto con un campo llamado **token**. Este es el token de autenticación que usaremos para poder acceder a los servicios relacionados con los eventos, y debemos guardarlo en **LocalStorage** (con la clave "**token**", tiene que ser la misma que la configurada en las peticiones al servidor).

   Si el login falla, el servidor devuelve un error **401** (*Not Authorized*). Pero el error no lo vamos a gestionar aquí, sino desde donde hemos llamado a este método.

- register

Este método llamará por POST al servicio ${SERVER}/auth/register y enviará los datos del usuario que, como se puede ver en el formulario, son los siguientes:

```
{
   "username": "Nombre del usuario",
   "email": "Correo del usuario",
   "password": "Contraseña",
   "avatar": "Imagen del usuario en Base64 "
}
```

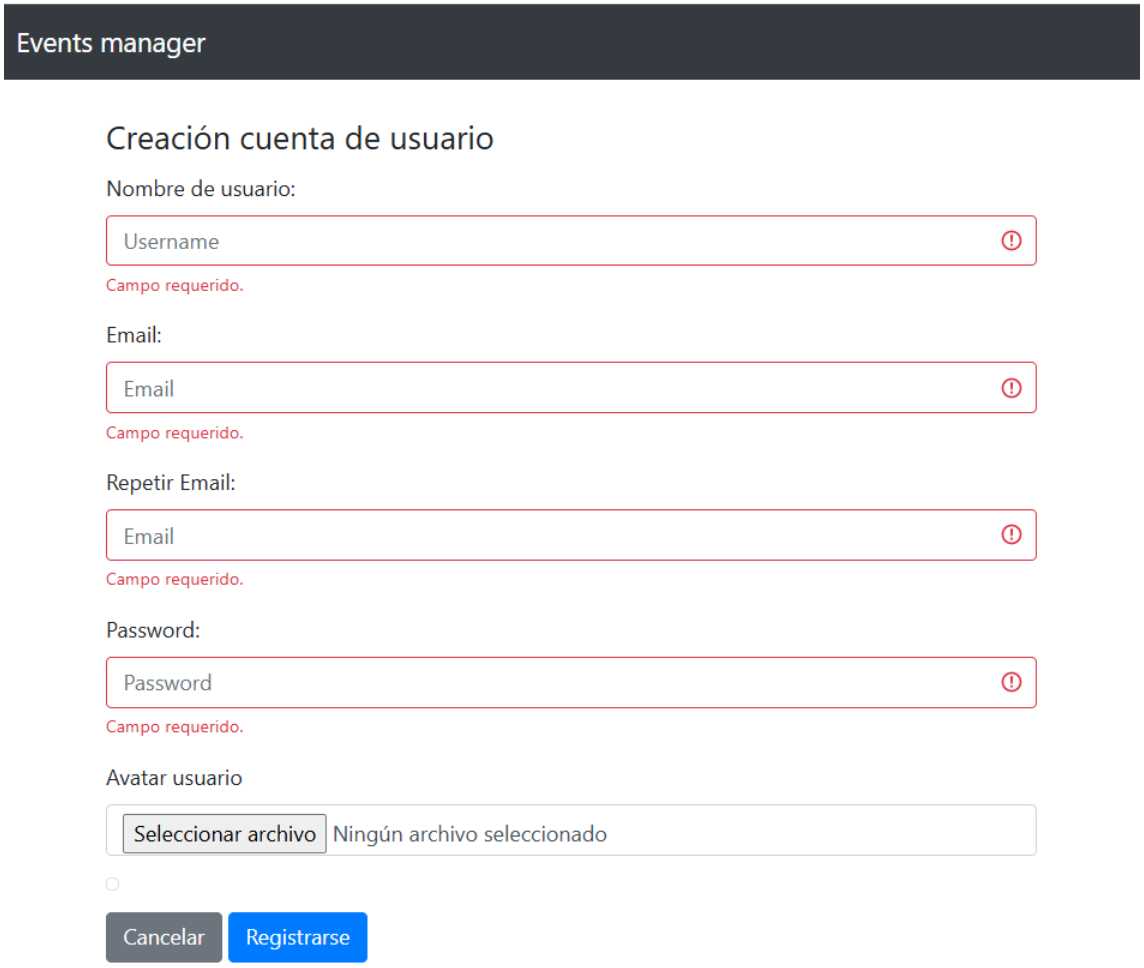
- checkToken

Este método comprobará si el token almacenado es válido. Para ello lo único que haremos será una llamada GET a ${SERVER}/auth/validate. El servicio devuelve `{ok:true}` si el token es válido, o un error 401 si falla.

- logout

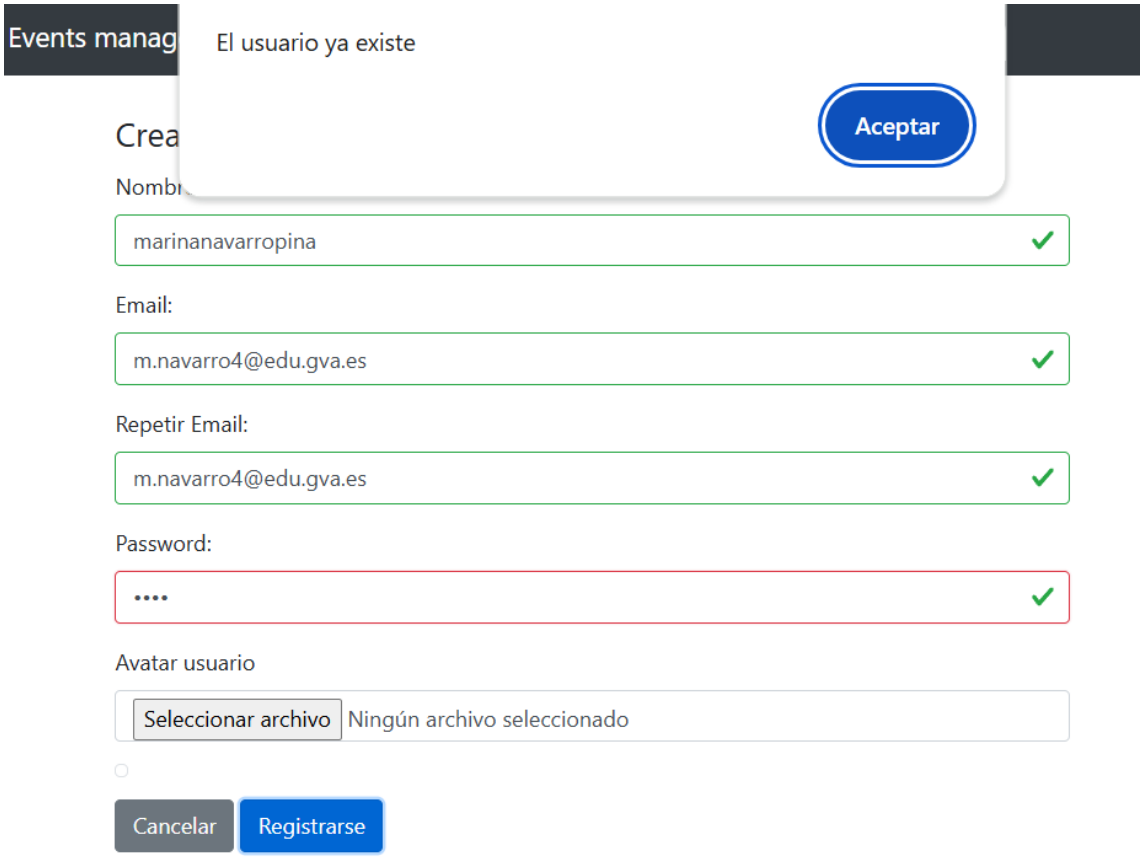
Este método simplemente borrará el token de LocalStorage.

3. Crear **register.js** para gestionar el formulario de registro de nuevo usuario (register.html). El archivo register.html identifica los campos requeridos para hacer el registro. (2,5 puntos)



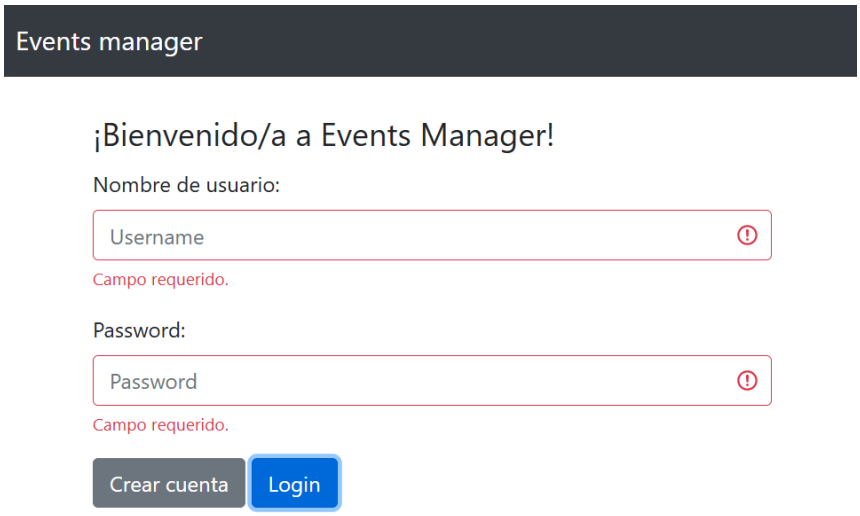
Se ha de validar que ningún campo esté vacío y que los 2 correos del formulario sean iguales antes de enviarlo.

Si todo ha ido bien devolverá un código **201**, y si ha fallado devolverá un error; por lo que se informará al usuario.



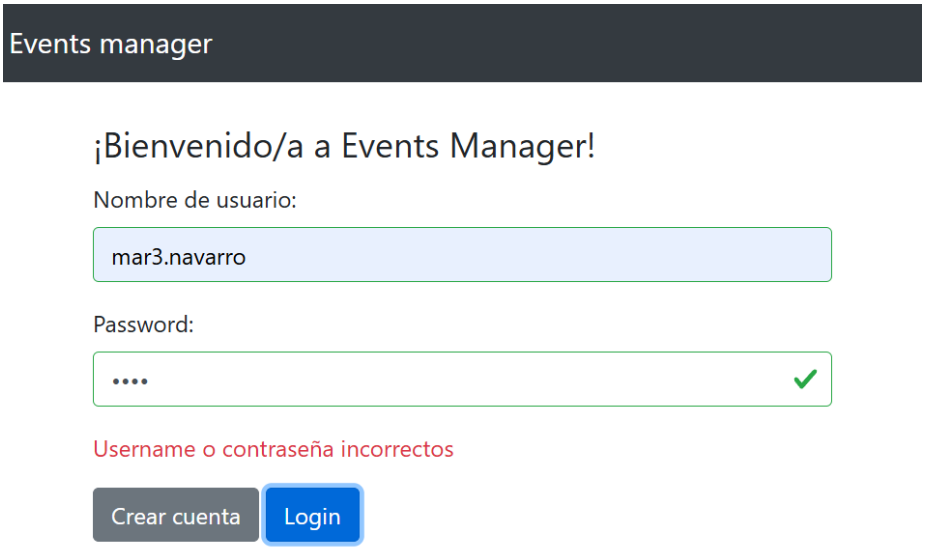
Si el registro es correcto, se debe redirigir a la página de login.html.

4. Crea **login.js** para gestionar la página de autenticación de usuarios (login.html). Este archivo será el punto de entrada a la aplicación. Por otro lado, el archivo login.html identifica los campos requeridos para que el usuario pueda realizar login en la aplicación. (1,5 puntos)



En el mismo se procesará el envío de datos del formulario de autenticación y se llamará al método **login** de la *clase Auth* pasándole el `username` y el `password` del formulario (no hace falta validación).

Si la respuesta del servicio de login es correcta, redirige a index.html. Si la respuesta no es correcta, mostrará el error quedándose en login.html.



5. Verificación de token: (1 punto) - Puede dar error, cambiar a chrome

   Cuando se cargue la página login.html, se ha de realizar una llamada a Auth.checkToken para comprobar si el usuario dispone de un token de autenticación válido. Si devuelve una respuesta sin error (el usuario está autenticado), se ha de redirigir a index.html.

   En las páginas index.html y add-evento.html, se ha de realizar la operación contraria. Si al llamar a Auth.checkToken da un error, significa que no se ha iniciado sesión y se debe redirigir al login.

6. Cierre de sesión: (0,5 puntos)

Añade a las páginas index.html y add-evento.html un enlace en el menú para cerrar sesión. En ambas páginas debes gestionar el click en dicho enlace, llamar a Auth.logout() y redirigir a la página de login.html.

Se tendrá en cuenta:

- Además de la realización de los archivos solicitados y las consideraciones descritas, la estructuración y documentación del código. (1 punto)