Nodepop Frontend

Información de contacto del profesor:

Alberto Casero
alberto@kasfactory.net
Twitter: @KasAppeal

En la práctica del módulo anterior (Backend con Node) se pedía desarrollar un backend de una aplicación de gestión de anuncios. Bien, es la hora de crear el frontend de esa aplicación.

Requisitos mínimos

Este frontend deberá contar con las siguientes páginas y funcionalidades:

Página de listado de anuncios:

- En la página principal encontraremos un listado de anuncios. Cada anuncio presentará una foto (si tiene), nombre, precio y si es compra o venta.
    - Los anuncios deberán cargarse desde el backend (se detallará más adelante).
    - En este listado de anuncios se deberán gestionar todos los estados de interfaz correctamente: vacío (no hay anuncios), error (cuando se produce un error al cargar los anuncios), carga (mientras se cargan los anuncios desde el backend) y éxito (cuando se han recuperado los anuncios y están listos para ser mostrados).
- Al pulsar sobre un anuncio, se deberá enviar al usuario a la página de **detalle de un anuncio.**
- Si el usuario está autenticado, se deberá permitir al usuario crear un anuncio que, al pulsarlo, deberá llevar a la **página para crear un anuncio.**

Página de detalle de un anuncio:

- El detalle de un anuncio deberá mostrar foto (si tiene), nombre, precio y si es compra o venta.
- En este detalle de anuncio se deberá gestionar todos los estados de interfaz correctamente: vacío (no existe el anuncio), error (cuando se produce un error al cargar la información del anuncio), carga (mientras se cargan la información del anuncio desde el backend) y éxito (cuando se han recuperado la información del anuncio y está listo para ser mostrado).
- Si el usuario está autenticado y el anuncio le pertenece, deberá además mostrar un botón que permita eliminar el anuncio (aunque antes de eliminarlo, deberá confirmar con el usuario si realmente quiere eliminar o no el anuncio).

Página para crear un anuncio:

- En la página para crear un anuncio se deberá mostrar al usuario un formulario con los siguientes campos:
    - Foto (opcional): permitirá subir una foto del anuncio
    - Nombre (obligatorio): nombre del anuncio
    - Precio (obligatorio): precio del anuncio
    - Compra/venta (obligatorio): indica si el anuncio se trata de una compra o de venta

- Cuando el usuario envíe el formulario, deberá enviar al backend una petición para guardar el anuncio.
- Se deberá gestionar todos los estados de interfaz correctamente: error (cuando se produce un error al guardar la información del anuncio), carga (mientras se guarda la información del anuncio en el backend) y éxito (cuando se han guardado correctamente la información del anuncio).

Página de login:

- La página de login deberá mostrar un formulario solicitando el nombre de usuario y contraseña.
- Cuando el usuario envíe el formulario, deberá autenticar al usuario contra el backend para obtener un token JWT que será utilizado en las siguientes comunicaciones con el backend para autenticar al usuario.
- Se deberá gestionar todos los estados de interfaz correctamente: carga, error y éxito.

Página de registro:

- Muy parecida a la de login. Deberá mostrar un formulario solicitando el nombre de usuario y contraseña.
- Cuando el usuario envíe el formulario, deberá registrar al usuario en el backend.
- Se deberá gestionar todos los estados de interfaz correctamente: carga, error y éxito.

Requisitos opcionales

Si te ha sabido a poco la práctica, te animo a que intentes implementar las siguientes funcionalidades (puedes elegir las que quieras):

- Gestionar la paginación de anuncios en el listado, ya que por defecto json-server sólo devuelve 10 elementos.
- Implementar un buscador de anuncios en el listado.
- Permitir editar un anuncio, sólo si el usuario autenticado es el propietario del anuncio.
- Permitir el filtrado de anuncios usando tags. Por lo que en el formulario de anuncio deberán poder incluirse tags de los mismos. Estos tags inicialmente pueden ser estáticos (siempre los mismos).
- Unido al anterior, hacer que los tags sean dinámicos.

Restricciones

Todo el desarrollo deberá realizarse usando Vanilla JavaScript, es decir, no se permite el uso de ninguna librería o framework. Todo el código JavaScript deberá ser creado por ti.

Sí está permitido el uso de librerías o frameworks CSS para acelerar el proceso de maquetación.

Además, deberás proporcionar un archivo `db.json` para el backend con los datos de ejemplo para la corrección de la práctica (así no tengo que generar yo todo el juego de datos :D).

El backend

El backend a utilizar será sparrest.js, basado en json-server, el cual nos ofrece un completo API REST para simular un backend real.

Para hacerlo funcionar, únicamente hay que descargarse el código desde https://github.com/kasappeal/sparrest.js y, dentro de la carpeta donde se aloja el código, instalar las dependencias ejecutando:

```
npm i
```

Y para arrancar el servidor ejecutar:

```
npm start
```

Por defecto, arrancará el servidor en el puerto 8000, por lo que se podrá acceder a él a través de http://127.0.0.1:8000/

Este backend expone los siguientes endpoints:

- `POST /auth/register`: permite registrar un usuario. Recibe como parámetros username y password y devuelve si se ha podido o no registrar al usuario (no permite usuarios con el mismo username en el sistema).
- `POST /auth/login`: endpoint de autenticación. Recibe como parámetros username y password y devuelve un token JWT de autenticación.
- `POST /upload`: que permite la subida de archivos a través de un atributo file. Sólo se pueden subir archivos usando el formato multipart/form-data.
- En `/api/`:
    - Se encuentran los endpoints ofrecidos por json-server, por lo que se aconseja la lectura de su documentación.
    - Para usar los métodos POST, PUT o DELETE en cualquier subruta de `/api/`, será necesaria la autenticación usando token JWT.
    - Esta autenticación se realiza añadiendo a las peticiones HTTP una cabecera `Authorization: Bearer <token>`, donde `<token>` es el valor del token obtenido en el endpoint de login.