

Frontend Wallapop

Requisitos de la práctica

Esta práctica consiste en el desarrollo de una aplicación web similar a Wallapop. **Para el desarrollo de esta práctica no está permitido utilizar librerías o frameworks de JavaScript.** En cambio, sí está permitido utilizar utilidades de CSS externas como Tailwind, Bootstrap, etc para facilitar la maquetación y el **diseño.**

Además, deberás proporcionar un archivo db.json para el backend con los datos de ejemplo para la corrección de la práctica.

Como requisitos funcionales mínimos para aprobar la práctica, hay que implementar:

1. Listado de anuncios.

Cada anuncio debe mostrar su imagen(si tiene), nombre, descripción, precio y si es compra o venta. Los anuncios publicados deben obtenerse a través de un endpoint mencionado más adelante.

La pantalla de listado de los anuncios deberá 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, iremos a la pantalla de detalle de anuncio.

Si el usuario ha hecho login, hay que mostrar al usuario un botón que le permita acceder a la pantalla de creación de un anuncio.

2. Detalle de anuncio

La página de detalle de anuncio deberá mostrar foto (si tiene), nombre, descripción, 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 ha 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).



3. Creación de 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 producto.
- Nombre (obligatorio): nombre del producto.
- Descripción (obligatorio): descripción del producto.
- Precio (obligatorio): precio del producto.
- Compra/venta (obligatorio): indica si el anuncio se trata de una compra o una 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).



A esta pantalla sólo podremos acceder si estamos logados. En caso contrario, habrá que redireccionar al usuario a la página de listado de anuncios, informándole del motivo.

4. 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.



5. 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 [nuestro API](#) 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.

API REST de apoyo para la práctica

El que utilizaremos será [sparrest.js](#) (gracias a Alberto Casero por el currazo), proyecto basado en la utilidad [json-server](#), el cual nos ofrece un completo API REST para simular un backend real y adaptarse a las necesidades de esta práctica.

Para hacerlo funcionar, únicamente hay que clonar el repositorio de sparrest.js y, dentro de la carpeta donde se aloja el código, instalar las dependencias ejecutando el siguiente comando:

```
npm install
```

Una vez instaladas las dependencias, para arrancar el servidor ejecutamos:

```
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 API REST 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 mediante 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.