Práctica de fundamentos de React

Vamos a crear una aplicación de tipo dashboard que será la interfaz gráfica desde la que podremos gestionar el API de anuncios Nodepop.

Backend

Usaremos todos el siguiente proyecto como backend.

https://github.com/davidjj76/nodepop-api

Una vez en marcha, tendremos nuestro backend corriendo en el puerto 3001 (configurable via archivo .env). Tenéis disponible un swagger en la ruta **/swagger** donde podréis probar los diferentes endpoints y ver cómo pasar los datos en las peticiones.

En el backend tendremos disponibles los siguientes endpoints:

- **/api/auth/signup**
  - **POST:** Nos permite crear usuarios.
- **/api/auth/me**
  - **GET:** Nos devuelve la información del usuario autenticado
- **/api/auth/login**
  - **POST**: Devuelve un token de acceso cuando le pasamos un email y password de un usuario correctos.
- **/api/v1/adverts**
  - **GET**: Devuelve un listado de anuncios, con la posiblidad de aplicar filtros con la query que enviemos en la URL. Los filtros posibles son:
    - name=coche (que el nombre empiece por “coche”, sin importar MAY/MIN)
    - sale=true/false (si el anuncio es de compra o venta)
    - price=0-25000 (precio dentro del rango indicado)
    - tags=motor,work (que tenga todos los tags)
  - **POST**: Crea un anuncio.
- **/api/v1/adverts/tags**
  - **GET:** Devuelve el listado de tags disponibles.
- **/api/v1/adverts/:id**
  - **GET**: Devuelve un único anuncio por Id.
  - **DELETE**: Borra un anuncio por Id.

NOTA: Todos los endpoints bajo **/adverts** requieren que se envíe el token proporcionado en el endpoint de login. Se ha de enviar en la cabecera de la petición de la siguiente forma:

```
Header['Authorization'] = `Bearer ${token}`
```

Los datos del backend son persistidos en una base de datos sqlite en el directorio **/data** (de ese modo no os teneís que preocupar de crear bases de datos).
Las fotos subidas al backend son almacenadas en el directorio **/uploads** y servidas

por el backend cómo contenido estático en **/public** (la ruta pública de cada foto es almacenada en la base de datos).

Frontend

La aplicación frontend será una SPA (Single Page Application) desarrollada con React como librería principal. Podéis crear la aplicación con **create-react-app o vite** (o cualquier otra) para que no os tengáis que preocupar de la inicialización del proyecto.

En la aplicación se implementarán una serie de rutas (enrutado en el navegador) divididas en dos grupos: **Públicas y Protegidas.** En cada una de la rutas se renderizará un componente principal tal como se explica a continuación.

- **Públicas**: Accesibles para cualquier usuario.
  - **/login**: LoginPage
- **Protegidas**: Accesibles **SOLO** para usuarios autenticados. Cualquier acceso de un usuario no autenticado a cualquiera de estas rutas redireccionará a **/login**.
  - **/**: Redirecciona a **/adverts**
  - **/adverts**: AdvertsPage
  - **/adverts/:id**: AdvertPage
  - **/adverts/new**: NewAdvertPage
  - Para cualquier otra url que no coincida se creará un componente **NotFoundPage** que informará al usuario que la página solicitada no existe (la típica 404).

Funcionalidad de cada página-componente:

- **LoginPage:**
  - Formulario con inputs para recoger email y password del usuario.
  - Checkbox “Recordar contraseña” mediante el que indicaremos si guardamos en el localStorage el hecho de que hay un usuario logado, evitando tener que introducir credenciales en cada visita al sitio (pensad la información mínima que os interesea guardar).
- **AdvertsPage:**
  - Listado de anuncios. Cada anuncio presentará nombre, precio, si es compra o venta y los tags. No es necesario mostrar la foto en este listado.
  - Manejará el estado cuando no haya ningún anuncio de mostrar, con un enlace a la página de creación de anuncios.
  - Cada anuncio del listado tendrá un enlace al detalle del anuncio (ruta **/adverts/:id**).
  - Zona de filtros: Formulario con distintos inputs, donde podremos introducir los filtros que queremos aplicar sobre el listado.
    La idea es que a medida que vayamos eligiendo filtros se reduzca el número de anuncios mostrados, es decir, los anuncios filtrados mostrados deben cumplir todos los filtros elegidos.

Se puede implementar de manera que a cada interacción del usuario se vaya filtrando el resultado o bien al hacer click sobre un botón para aplicar el filtrado.
Habrá que implementar al menos 2 de estos filtros:
- Filtro por nombre (input tipo texto)
- Filtro compra/venta (input tipo radio o select con 'venta', 'compra', 'todos')
- Filtro por precio (input donde podremos seleccionar el rango de precios por el que queremos filtrar). Un slider o simplemente un par de inputs de tipo number para el precio mínimo y máximo.
- Filtro por tags (input donde podremos seleccionar uno o varios tags de los disponibles). Puede ser un input tipo select múltiple o varios checkboxes que permitan elegir varias opciones a la vez. Al aplicar el filtro se mostrarán los anuncios que contengan **todos** los tags elegidos.

- **Podemos manejar el filtrado de anuncios de dos formas (a elegir, aunque recomiendo la segunda para prácticar más con el estado de React)**.
  - Recoger los filtros a aplicar en el front y enviarlos a la petición al API para traer los anuncios ya filtrados desde el backend (una petición cada vez que se apliquen los filtros).
  - Traer los anuncios sin filtrar desde el backend, y aplicar el filtro en el frontend con lo que se haya recogido en el formulario de filtros (una única petición).

- **AdvertPage**:
  - Detalle del anuncio cuyo id es recogido de la URL. Mostrará la foto del anuncio o un placeholder en su lugar si no existe foto.
  - Si el anuncio no existe deberia redirigirnos al **NotFoundPage.**
  - Botón para poder borrar el anuncio. Antes de borrar mostar una confirmación al usuario (algo más elaborado que un window.confirm, jugando con el estado de React). Tras el borrado debería redireccionar al listado de anuncios.
- **NewAdvertPage:**
  - Formulario con **TODOS** los inputs necesarios para crear un nuevo anuncio:
    - Nombre
    - Compra / Venta
    - Tags disponibles.
    - Precio
    - Foto
  - Todos los campos, excepto la foto serán requeridos para crear un anuncio. Manejar estas validaciones con React, por ejemplo desabiltando el submit hasta pasar todas las validaciones.
  - Tras la creación del anuncio debería redireccionar a la página del anuncio.

- Además de estos componentes necesitaremos un componente visible cuando el usuario esté logeado desde el que podamos hacer **logout** (un botón por ejemplo, a poder ser possible con confirmación, pensando en reusar lo que hayamos hecho en la confirmación de borrado)**.**
- Las rutas de /adverts y /adverts/new deben de estar accesibles fácilmente mediante enlaces de navegación (Link o NavLink).

Otras cosas

A tener en cuenta.

- **Estilos**: No es necesario una aplicación super mega impactante en cuanto a lo visual, simplemente que funcione y que las cosas esté colocadas correctamente en la pantalla.
  - Podéis usar la técnica de estilado que más os guste, CSS puro, TailwindCSS, atributo style, SCSS, CSS modules, CSS in JS como Styled Components (u otros) o una mezcla de cosas.
  - Podéis usar alguna librería de componentes puramente visuales si creeis que os puede quitar trabajo a la hora de dar un aspecto más uniforme a toda la aplicación (material UI, Chakra, Ant Desing...)
- **Librerías:** Intentad usar el mínimo de librerías posibles, React Router, axios están permitidas, pero por ejemplo no uséis librerías para el manejo de formularios (React Hook Form, o Formik), porque en estos momentos es preferible que comprendáis como funcionan las cosas en React.
- **Código:** Recomiendo usar un formateador (Prettier) de código por varias razones:
  - Formato uniforme por todo el código. El que corrige la práctica lo agradecerá .
  - No nos tenemos que preocupar al codificar por aspectos como indentaciones, saltos de linea, etc, el formateador lo hace por nosotros y escribiremos código más rápido.
  - Fácil de integrar en nuestro editor favorito mediante plugins. En equipos de desarrollo más grandes podemos incluso integrarlo en el proyecto y hacer que a cada commit se formetée automáticamente, asegurando que todos los miembros del equipo formatean igual independientemente de las configuraciones de sus editores.