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 que hemos desarrollado en los módulos anteriores.

Backend

Usaremos todos el siguiente proyecto como backend.

https://github.com/davidjj76/nodepop_web_avanzado

Este proyecto nos proporciona un API que contiene los siguientes endpoints (todos dentro de /apiv1):

- **/auth/login**
  - **POST**: Devuelve un token de acceso cuando le pasamos un email y password de un usuario correctos.
- **/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.
- **/adverts/tags**
  - **GET:** Devuelve el listado de tags disponibles.
- **/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 /auth/login. Se ha de enviar en la cabecera de la request de la siguiente forma:

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

Frontend

La aplicación frontend será una SPA (Single Page Application) desarrollada con React como librería principal. Podéis crear la aplicación con **create-react-app** 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
    - **/advert/:id**: AdvertPage
    - **/advert/new**: NewAdvertPage
    - Para cualquier otra url que no coincida se creará un componente **NotFoundPage** que informará al usuario que la página solicitada no existe.

Funcionalidad de cada página-componente:

- **LoginPage:**
    - Formulario con inputs para recoger email y password del usuario.
    - Checkbox "Recordar contraseña" mediante el que indicaremos que guardamos en el localStorage los datos de la session de usuario, evitando tener que introducir credenciales en cada visita la sitio.
- **AdvertsPage:**
    - Listado de anuncios. Cada anuncio presentará nombre, precio, si es compra o venta y los tags. No mostrará 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 **/advert/:id**).
    - Zona de filtros: Formulario con distintos inputs, donde podremos introducir los filtros que queremos aplicar en la llamada al API para obtener los anuncios.
        - Filtro por nombre (input tipo texto)
        - Filtro compra/venta (input tipo radio 'venta', 'compra', 'todos')
        - Filtro por precio (input donde podremos seleccionar los rangos de precio por los que queremos filtrar). Podeis usar un componente como este. https://github.com/react-component/slider
        - Filtro por tags (input donde podremos seleccionar uno o varios tags de los disponibles). El filtro incluirá todos los tags seleccionados.
        - **EXTRA**: Estaría bien que la aplicación "recordase" las preferencias de filtrado del usuario, de modo que cada vez que se entre en esta ruta estuviesen ya marcados los últimos filtros aplicados y con ellos se realizase la petición al API. Estas preferencias deberían permanecer guardadas aunque cerremos el navegador.
- **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. Estaría bien mostar una confirmación antes de borrar. 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 inputs, excepto la foto, son requeridos para poder crear el 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.**
- 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, atributo style, SCSS, CSS modules, CSS in JS como Styled Components (u otros) o una mezcla de cosas.
    - Podéis usar una librería de componentes si creeis que os puede quitar trabajo a la hora de dar un aspecto más uniforme a toda la aplicación. Hay muchas, como Semantic UI, React Bootstrap, Ant…
- **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.