Trabajo realizado por Carlos Maranhas, Adrián Rafael de la Cruz y Rubén Navarro S2DAW

Practica 9.3: Proyecto SPA usando React, React Router y acceso a una API de datos

Este proyecto trata del uso de una api, react y la distribución de react-router.

Para empezar este trabajo, primero debemos tener claro que vamos a necesitar para poder crearla, es decir, los componentes. Para saber esto, podemos crear un prototipo gracias a Figma.

Podemos observar que el objetivo de la app es tener un buscador, que al buscar salgan tres card por fila con las canciones relacionadas con la búsqueda, que tienen un botón para guardarlas en la columna de la izquierda, la cual dispone de botones para eliminar las canciones. Y, por último, al hacer clic en la canción, artista o álbum, que salga información varia.

Si comparamos el prototipo con la app final, se pueden apreciar las mejoras ya que, a la hora de ir realizando la app, se han ido incorporando nuevas ideas, como por ejemplo que las card tengan animación al pasar por encima el puntero.

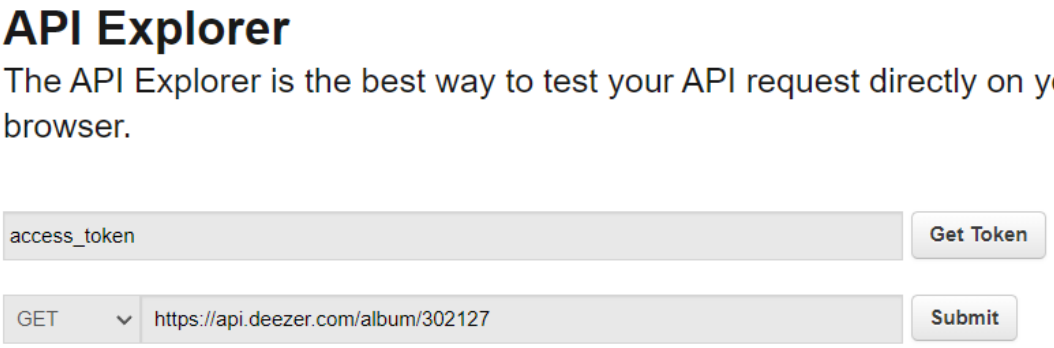
ENLACE AL REPOSITORIO DONDE SE ENCUENTRA EL PROYECTO

https://github.com/Carlistos10/API-DEEZER-REACT.git

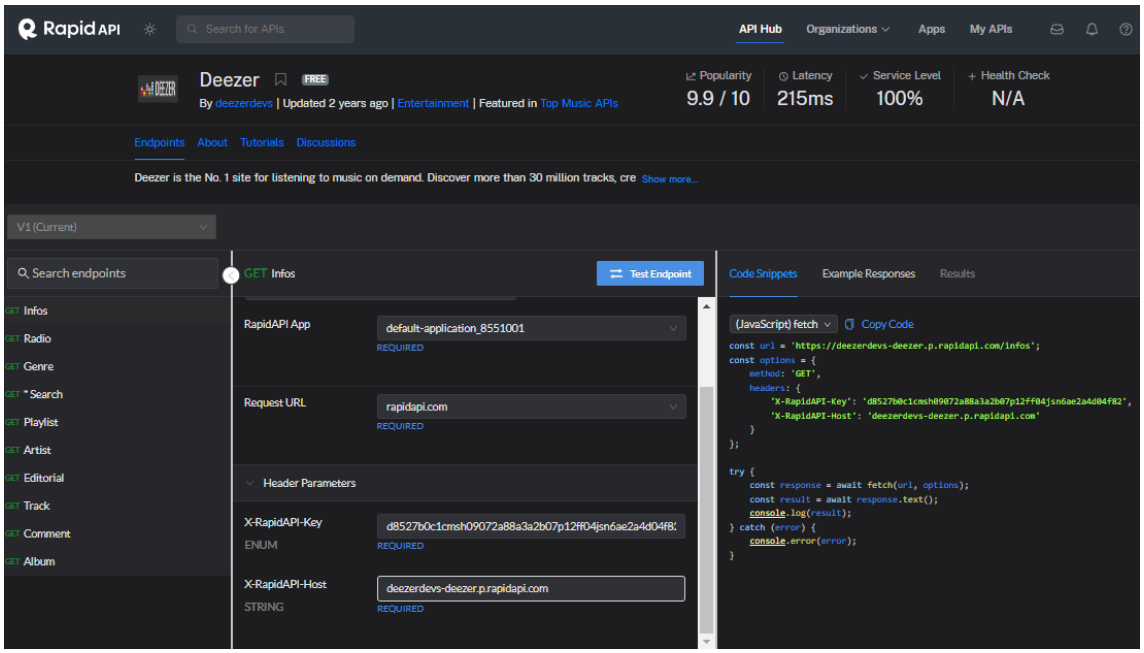
API ESCOGIDA

La api escogida en nuestro grupo es la api de DEEZER, que es de una aplicación de música.

Para poder acceder a esta api, realizamos la búsqueda en internet del api correspondiente, pero la página web en la que se encontraba la api, no nos facilitaba el token para poder usarla, por lo que tuvimos que buscar otra opción.



Realizando búsquedas en internet de como usar las apis, nos encontramos la pagina web de rapidapi, la cual es una página dedicada al uso y aprendizaje de las apis, por lo que decidimos buscar la api de deezer. Encontramos la api que necesitábamos y esta página si que nos proporcionaba una api-key y un api-host.



En esta página aprendimos a usar la api, ya que los endpoints de la izquierda son los apartados que tiene esta api, los cuales son los encargados de proporcionarnos los datos solicitados.

Gracias a estos endpoints, descubrimos que el único apartado que permite string (texto), era el search, el resto funcionan con id, los cuales se obtienen a través del search.

Gracias a esto, nos dimos cuenta que la clave de esta api son los id, ya que gracias a ellos obtenemos toda la información de las canciones, álbumes, artistas, playlists…

Componentes de la aplicación

La aplicación cumple con los 6 componentes requeridos, que son:

- Álbum:

album.jsx

src > components > album.jsx > ...

```jsx
import React from 'react'
import { useParams, Link } from 'react-router-dom';

export const Album = () => {
    const { album } = useParams(); // Obtén los parámetros de la URL


    return (
        <div>
            <h1>Detalles del álbum: {album}</h1>
            <p>Este album pertenece a un gran artista, el cual dedica
            <Link to={`/`}>
                <span>Volver</span>
            </Link>
        </div>
    );
}

```

Este componente nos presenta información varia del álbum, al no funcionarnos el enviar los datos de la api a los links de REACT-ROUTER, nos tuvimos que ver obligados a poner a mano alguna información.

- Artista:

artista.jsx

src > components > artista.jsx > ...

```jsx
import React from 'react';
import { useParams, Link } from 'react-router-dom';

export const Artista = () => {
    const { artist } = useParams(); // Obtén los parámetros de la URL


    return (
        <div>
            <h1>Detalles del artista: {artist}</h1>
            <p>Este artista tiene una gran variedad de albumes, y ha co
            <Link to={`/`}>
                <span>Volver</span>
            </Link>
        </div>
    );
};

```

Nos ofrece información del artista, y nos ocurre el mismo problema que con el álbum.

- Canción:

```jsx
// cancion.jsx
// src > components > cancion.jsx > default
import React from 'react';
import { useParams, Link } from 'react-router-dom';

const Cancion = () => {
  const { artist, album, cancion } = useParams(); // Obtén los parámetros de la URL

  return (
    <div>
      <h1>Detalles de la Canción: {cancion}</h1>
      <p>Artista: {artist}</p>
      <p>Álbum: {album}</p>
      <Link to={`/`}>
        <span>Volver</span>
      </Link>
    </div>
  );
};

export default Cancion;

```

Observamos información varia de la canción seleccionada, en este caso, aunque el problema persistía, conseguimos recopilar un poco más de información.

- Búsqueda

```jsx
export const Busqueda = ({ playList }) => {
  async function fetchData(searchTerm) {
    if (!searchTerm) return [];

    const ajustesBusqueda = {
      "async": true,
      "crossDomain": true,
      "url": `https://deezerdevs-deezer.p.rapidapi.com/search?q=${searchTerm}`,
      "method": "GET",
      "headers": {
        "x-rapidapi-host": "deezerdevs-deezer.p.rapidapi.com",
        "x-rapidapi-key": "d8527b0c1cmsh09072a88a3a2b07p12ff04jsn6ae2a4d04f82"
      }
    }

    return fetch(ajustesBusqueda.url, {
      method: ajustesBusqueda.method,
      headers: ajustesBusqueda.headers
    })
      .then(response => response.json())
      .then(resultadoBusqueda => {
        console.log("Array de la busqueda: ");
        console.log(resultadoBusqueda);
        return resultadoBusqueda.data || [];
      })
      .catch(error => {
        console.error('Error al realizar la búsqueda:', error);
```

En este componente obtenemos toda la información de la api para poder repartirla por los diferentes componentes.

- Eliminar

```jsx
queda.jsx   eliminar.jsx
omponents > eliminar.jsx > ...
import React from 'react';
import './eliminar.css'

export const Eliminar = ({ onEliminar }) => {
    return (
        <button onClick={onEliminar} className='boton'>Eliminar</button>
    );
};
```

Elimina las canciones de la playlists

- Tarjeta:

```jsx
queda.jsx   tarjeta.jsx
omponents > tarjeta.jsx > Tarjeta
export const Tarjeta = ({ data }) => {
  const { title, album, artist } = data;

  return (
    <div className="container">
      <div className="card">
        <Link to={`/detalleCancion/${artist.name}/${album.title}/${title}`}>
          <div className="imgBx">
            <img src={album.cover_medium} alt='' />
          </div>
        </Link>
        <div className="contentBx">
          <h2>{title}</h2>
          <div className="size">
            <h3>Artista :</h3>
            <Link to={`/detalleArtista/${artist.name}`}>
              <span>{artist.name}</span>
            </Link>
          </div>
          <div className="color">
            <h3>Álbum :</h3>
            <Link to={`/detalleAlbum/${album.title}`}>
              <span>{album.title}</span>
            </Link>
          </div>
        </div>
      </div>
```

Ofrece la información de la búsqueda en forma de card, en las que, para obtener mas información de las canciones, usamos los LINKS de REACT-ROUTER.

RUTA CREADAS:

busqueda.jsx | tarjeta.jsx | App.js

App.js > ...

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { Busqueda } from './components/busqueda';
import Cancion from './components/cancion';
import { Artista } from './components/artista';
import { Album } from './components/album';


function App() {
  return (
    <Router>
      <div>
        <Routes>
          <Route exact path="/" element={<Busqueda />} />
          <Route path="/detalleCancion/:artist/:album/:cancion" element={<Cancion />} />
          <Route path="/detalleArtista/:artist" element={<Artista />} />
          <Route path="/detalleAlbum/:album" element={<Album />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;

```

El archivo App.js contiene todas las rutas necesarias para el máximo rendimiento de la app, en ella podemos encontrar la ruta en las que se ofrece mas información de los artistas, álbumes o canciones, y una ultima que te lleva de vuelta a la ruta principal de la app.