ALCYON
IT Servicios

PARTE VIII: Rutas

ALCYON
IT Servicios

INTRODUCCIÓN

En este documento veremos:

- Crear rutas para cargar componentes
- Crear la barra de navegación
- Pasar parámetros por las rutas

ALCYON
IT Servicios

INTRODUCCIÓN

Creación del proyecto

Crearemos un nuevo proyecto desde el cmd o el terminal de VSC:

create-react-app routing

Preparar el entorno

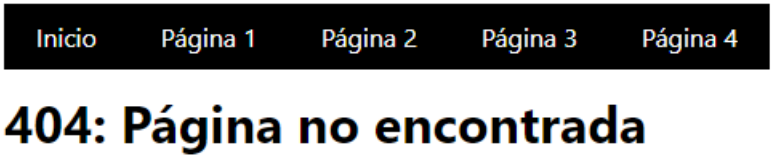
Vamos a editar el fichero **src/App.css** y borraremos el contenido

Editamos el fichero **src/App.js** y vaciamos el contenido del return de la función App()

CREAR LOS COMPONENTES

Crear componentes

Para este ejercicio crearemos, a parte del componente principal, cinco componentes (uno para cada una de las páginas de las que constará el ejercicio), un componente extra para mostrarlo en caso que la página a mostrar no exista (error 404) y un componente para la barra de navegación

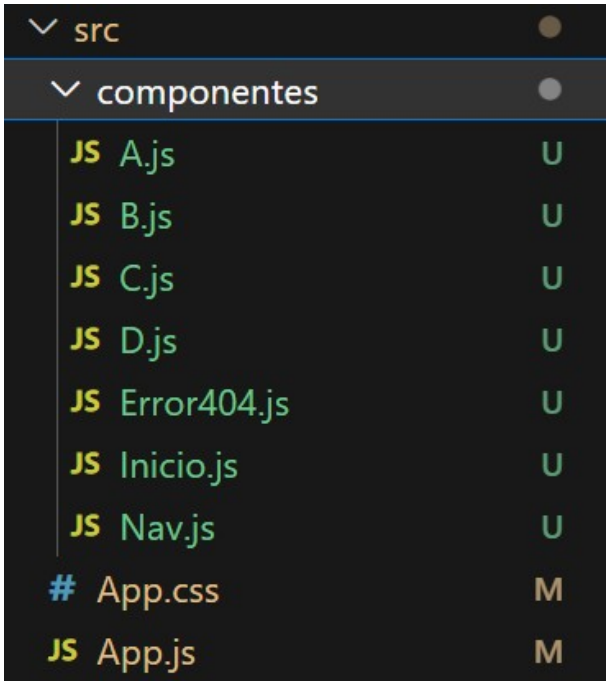


CREAR LOS COMPONENTES

Objetivo del ejercicio

El objetivo del ejercicio es comprender como funciona el enrutado de React creando una barra de navegación y cinco páginas que se cargarán al pulsar sobre cada una de las opciones de enlace de la barra

La estructura de carpetas que necesitaremos será la siguiente:



Inicio.js y **A.js** hasta **D.js** son las 5 páginas de la app

Error404.js es la página a mostrar en caso de error 404

Nav.js es el componente para la barra de navegación

CREAR LOS COMPONENTES

Estructura componentes Inicio.js, A.js, B.js, C.js y D.js

Los cinco componentes que corresponden a las cinco páginas tendrán la siguiente y sencillísima estructura:

```
return (
    <div>
      <h1>Página A</h1>
    </div>
)
```

NOTA: En cada componente substituimos el contenido del título por el texto correspondiente a cada página

ALCYON
IT Servicios

CREAR LOS COMPONENTES

Estructura componente Error404.js

El componente a mostrar en caso que la página a cargar no exista tendrá la siguiente estructura:

```
return (
    <div>
      <h1>404: Página no encontrada</h1>
    </div>
)
```

ALCYON
IT Servicios

INSTALAR EL ENRUTADO DE ANGULAR

Necesitamos instalar la librería para el enrutado de los componentes. Para ello, desde la linea de comandos o desde el terminal de VSC tecleamos:

```
npm i react-router-dom
```

Que nos instalará todas las funcionalidades del *React Router DOM* que necesitaremos para crear todas las rutas a cada uno de los componentes que hemos creado antes

<BrowserRouter>	→ Etiqueta que envolverá todo el sistema de enrutado
<Routes>	→ Etiqueta que agrupará todas las rutas del ejercicio
<Route>	→ Cada una de las rutas que se agrupan en <Routes>
<NavLink>	→ Cada uno de los enlaces de la barra de navegación
<Link>	→ Cada uno de los enlaces de fuera de la barra de navegación
useParams()	→ Para enviar parámetros por la url

ALCYON
IT Servicios

CONFIGURAR EL ENRUTADO

Componente principal App.js

Vamos a definir todas las rutas de la aplicación dentro del componente principal **App.js**.

Para ello necesitaremos importar, por una parte, todos los componentes que vamos a asociar a cada una de las rutas:

```
import A from './componentes/A'
import B from './componentes/B'
import C from './componentes/C'
import D from './componentes/D'
import Inicio from './componentes/Inicio'
```

Y, todos las librerías del *React Router DOM* que vamos a necesitar:

```
import {BrowserRouter, Route, Routes, Navigate} from 'react-router-dom'
```

ALCYON
IT Servicios

CONFIGURAR EL ENRUTADO

Componente principal App.js

Vamos a crear ahora todos los elementos del enrutado:

```
<div className="App">
    <BrowserRouter>
      <Routes>
        <Route path='/' element={<Inicio/>}/>
        <Route path='/pagina1' element={<A/>}/>
        <Route path='/pagina2' element={<B/>}/>
        <Route path='/pagina3' element={<C/>}/>
        <Route path='/pagina4' element={<D/>}/>
      </Routes>
    </BrowserRouter>
</div>
```

NOTA: vemos el uso de las etiquetas anteriores para englobar todo el enrutado (<BrowserRouter>), el grupo de rutas (<Routes>) y cada una de las rutas que cargarán un componente (<Route>)

NOTA: En cada etiqueta <Route> indicamos la ruta de la url asociada y el componente a cargar

CONFIGURAR EL ENRUTADO

Componente principal App.js

Si probamos la aplicación tal como está ahora veremos que en el navegador veremos el contenido asociado a la ruta “/” que corresponde con el componente **A.js**

Página de Inicio

Y si en la url indicamos cada una de las rutas deberíamos ver como se carga el componente asociado a cada una de ellas, Ejemplo:

http://localhost:3000/pagina2

Página B

ALCYON
IT Servicios

ERROR 404

Componente principal App.js

¿Qué pasa si indicamos una página que no existe. Por ejemplo **localhost:3000/pagina9**?

Pues que veremos una página en blanco. Para evitarlo vamos a crear una nueva ruta que asociaremos al componente **Error404.js**

```
<Route path='*' element={<Error404/>}/>
```

Es importante que esta ruta sea la última dentro del grupo <Routes> ya que es la que se va a ejecutar si no existe ninguna de las anteriores

404: Página no encontrada

REDIRECCIONAMIENTO

Componente principal App.js

Vamos a introducir otra mejora consistente en redireccionar a una ruta existente cuando el usuario introduce por la url una ruta que no tenemos en el grupo de rutas válidas

Por ejemplo, la ruta "**/**" corresponde al componente Inicio.js. Si el usuario introduce en la url "**/inicio**" veremos como se carga el componente de **error404**.

A veces, por temas de usabilidad, podemos hacer que esta ruta también se utilice para cargar el componente de inicio y la forma más sencilla de hacerlo es mediante el redireccionamiento

Dentro del grupo <Routes> añadimos:

<Route path='/inicio' element={<**Navigate** to="/"/>}/>

ALCYON
IT Servicios

BARRA DE NAVEGACION

Componente componentes/Nav.js

Obviamente nos falta una barra de navegación para que el usuario, al pulsar sobre cada uno de los enlaces, pueda cargar el componente correspondiente.

Para ello vamos a crear dentro de **componentes** un fichero **Nav.js** donde crearemos la barra de navegación con los correspondientes enlaces (que en angular no serán la etiqueta <a> sino la etiqueta <NavLink>

El primer paso es importar NavLink

```
import {NavLink} from 'react-router-dom'
```

ALCYON
IT Servicios

BARRA DE NAVEGACION

Componente componentes/Nav.js

Y ahora construimos nuestra barra de navegación:

```
<nav>
    <NavLink to={'/'}>Inicio</NavLink>
    <NavLink to={'/pagina1'}>Página 1</NavLink>
    <NavLink to={'/pagina2'}>Página 2</NavLink>
    <NavLink to={'/pagina3'}>Página 3</NavLink>
    <NavLink to={'/pagina4'}>Página 4</NavLink>
</nav>
```

Y un poco de css en **App.css**

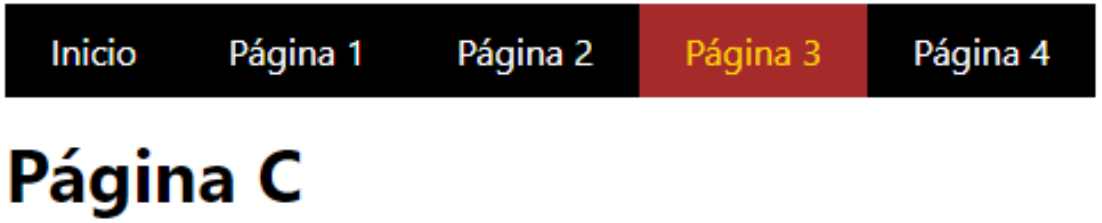
```
nav a {text-decoration: none; color: white; background-color: black; padding: 10px 20px;}

nav a:hover {background-color: brown;}
```

BARRA DE NAVEGACION

Componente principal App.js

Si probamos, vemos que todavía no funciona, ya que nos falta incorporar este componente **Nav.js** en el fichero principal **App.js**:

```
<BrowserRouter>
    <Nav/>
    <Routes>
      .../...
    </Routes>
</BrowserRouter>
```



NOTA: Que no se nos olvide importar el componente: import Nav from './componentes/Nav';

OPCION ACTIVA EN NAVEGACION

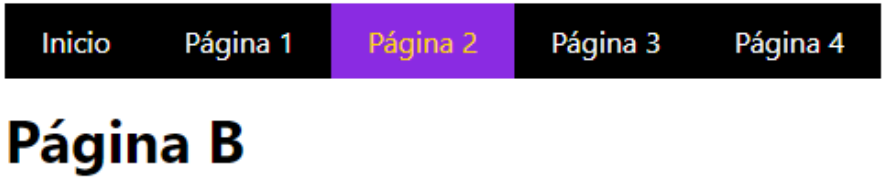
Componente Nav.js

Vamos a añadir una mejora de usabilidad consistente en que la opción del menú seleccionada se mantenga de un color distinto al resto.

Para ello, sólo, sólo tenemos que añadir en el fichero **App.css** una clase con las propiedades que queremos asignar a la opción activa de la barra de navegación:

```
.active {color:gold; background-color: blueviolet;}
```

Y no hay que hacer nada más. Vemos como React añadirá automáticamente esta clase a la opción activa:



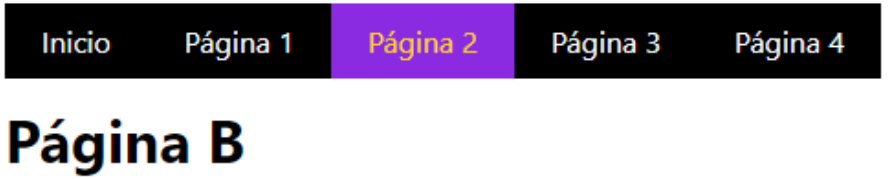
OPCION ACTIVA EN NAVEGACION

Componente Nav.js

Sólo para versiones de React Router DOM anteriores a la 6

En versiones anteriores a la 6 tendremos que añadir manualmente la clase active a la correspondiente ruta activa utilizando esta sintaxis dentro de cada una de las etiquetas NavLink del fichero Nav.js

```
<NavLink className={({isActive})=> (isActive ? "activado" : null)}
to={'/'}>Inicio</NavLink>
```

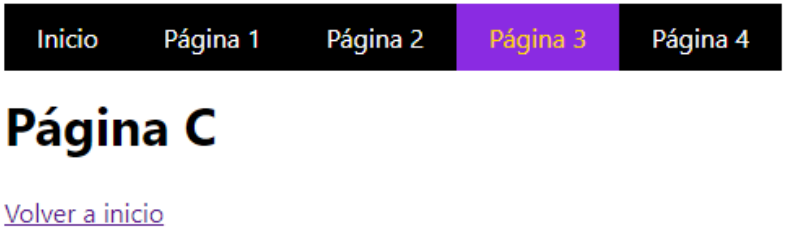


ENLACES FUERA DE LA BARRA NAV

Componente C.js

Vamos a modificar el componente **C.js** correspondiente a la ruta '**/pagina3**' para añadir un enlace que nos lleve a la página de inicio pero situado fuera de la barra de navegación. Para ello utilizaremos la etiqueta <Link>

```
<div>
    <h1>Página C</h1>
    <p>
      <Link to="/">Volver a inicio</Link>
    </p>
</div>
```



Qué tendremos que importar con:

```
import { Link } from 'react-router-dom'
```

ALCYON
IT Servicios

ENVIO DE PARAMETROS POR LAS RUTAS

Componente Nav.js

Vamos a ver como enviar parámetros por las rutas. Algo muy habitual en aplicaciones SPA.

Modificaremos el componente de la barra de navegación **Nav.js** para modificar la ruta que hemos asociado al enlace de la página 4 (que carga el componente **D.js**) y pasarle un dato como parámetro en la ruta:

```
<NavLink to={'/pagina4/99'}>Página 4</NavLink>
```

ENVIO DE PARAMETROS POR LAS RUTAS

Componente App.js

Tendremos que modificar el grupo de rutas del componente principal **App.js** para indicar que la ruta de la página 4 asociada al componente **D.js** espera recibir un parámetro por la url:

~~<Route path='/pagina4' element={<D/>}/>~~

<Route path='/pagina4**/:id**' element={<D/>}/>

ENVIO DE PARAMETROS POR LAS RUTAS

Componente D.js

Ahora modificaremos el componente **D.js** para recoger el parámetro que llegará por la ruta y, para ello, utilizaremos el hook useParams

Primero importamos el nuevo hook a utilizar:

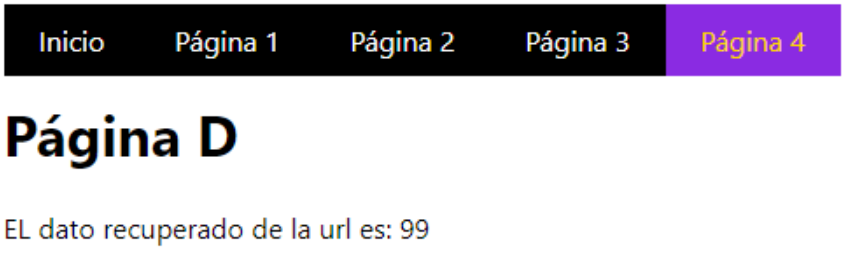
```
import { useParams } from 'react-router-dom'
```

En la función del componente usamos el hook para recuperar los datos que lleguen por la url:

```
const dato = useParams()
```

Y en el código JSX del componente mostramos el atributo id del objeto dato

```
<div>
    <h1>Página D</h1>
    <p>Dato recuperado es: {dato.id}</p>
</div>
```



ALCYON
IT Servicios

ENVIO DE PARAMETROS POR LAS RUTAS

Componente D.js

Si vemos en la consola el dato recuperado usando useParams veremos:

Vemos como el atributo id que hemos informado en la ruta del fichero **App.js** es el que nos llega como parte del objeto dato

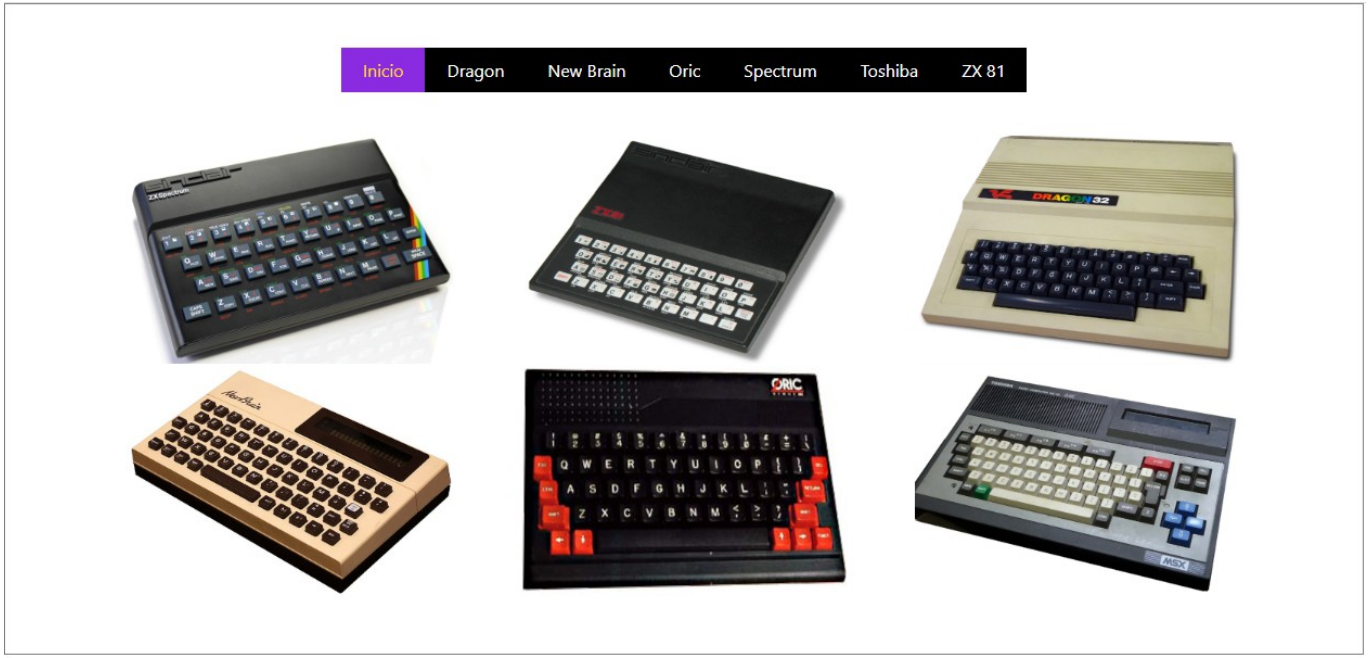
ALCYON
IT Servicios

ACTIVIDAD 8

ALCYON
IT Servicios

ACTIVIDAD

Vamos a realizar una actividad dividida en dos partes en donde pondremos en práctica el enrutado de React



Crearemos un nuevo proyecto y le añadimos la librería de enrutado:

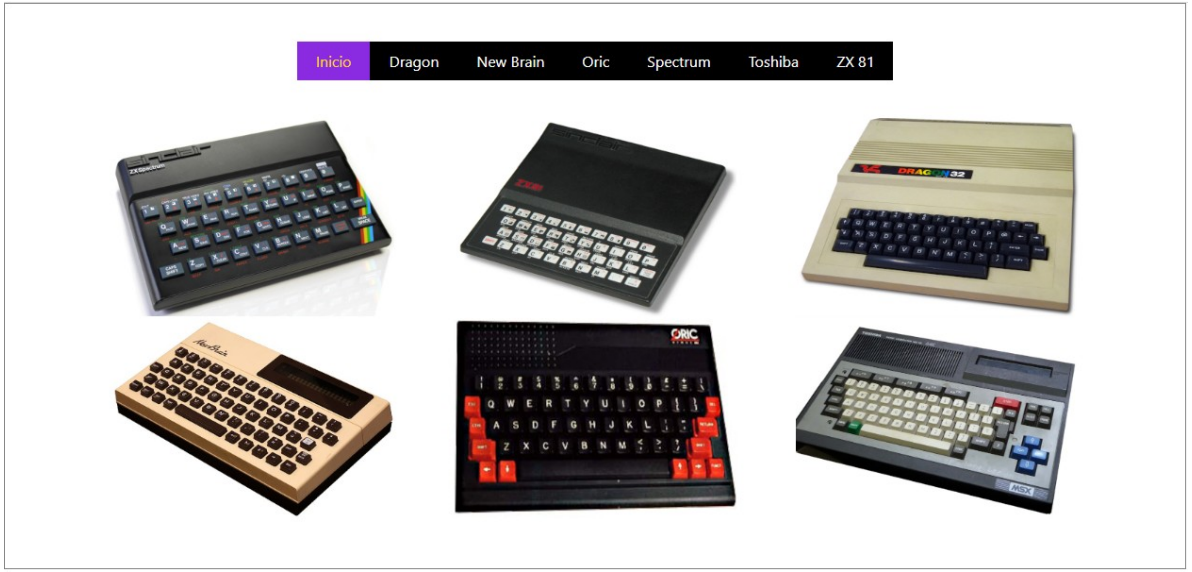
npm install react-router-dom

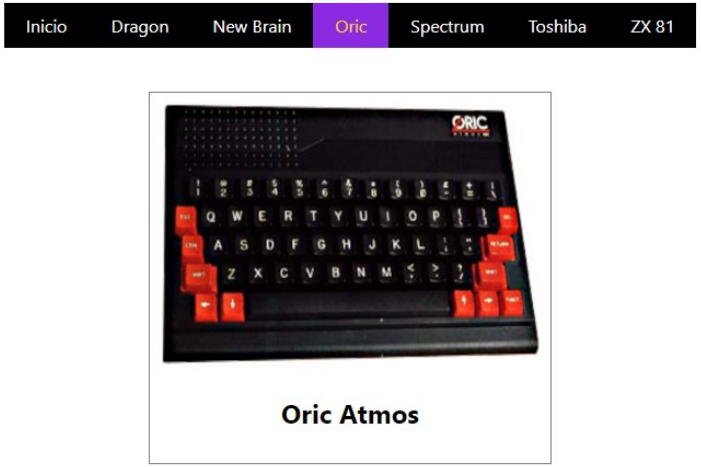
ACTIVIDAD

Actividad 8.1

En la primera parte vamos a crear una aplicación con una barra de navegación para enlazar con 6 componentes que mostrarán información básica de seis micro ordenadores vintage

Además, en la pantalla de inicio, se mostrarán las fotos de los 6 micros de forma que, al pulsar sobre cada una de ellas, se cargue también el componente que muestra información básica del micro seleccionado





ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.1: Crear array de micros

Utilizaremos el siguiente array con los micros a mostrar (en recursos)

```
const micros = [
    {
      path: 'spectrum',                    → nombre de la ruta a utilizar
      modelo: 'ZX Spectrum',               → nombre del micro
      imagen: 'sinclair_zx_spectrum.png'   → nombre de la imagen
    },
    {
      path: 'zx81',
      modelo: 'ZX 81',
      imagen: 'Sinclair_ZX81.png'
    },
    .../…
]
```

NOTA: Lo podemos incorporar directamente en el componente donde lo necesitaremos o bien crear un componente **Micros.js**, exportarlo y, posteriormente, importarlo desde el componente

ALCYON
IT Servicios

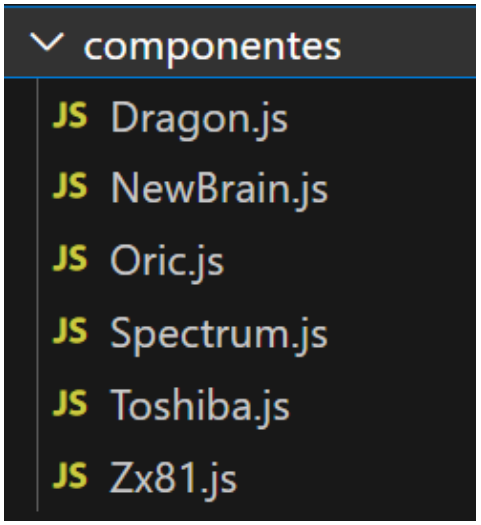
ACTIVIDAD

Actividad 8.1: Crear componentes

Vamos a crear seis componentes con la siguiente y sencilla estructura:

Ejemplo para el componente **Dragon.js**

```
<div className='ficha'>
    <img src='../img/dragon_32.png' alt="Dragon 32"/>
    <h2>Dragon 32</h2>
</div>
```



Utilizaremos también algo de css en **App.css**:

```
.ficha {width: 30%;margin:50px auto;
border: 1px solid grey;padding:10px}

img {width:100%}
```

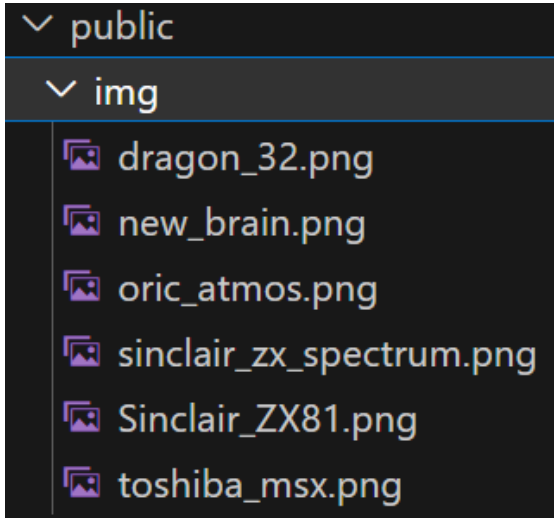
ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.1: Crear componentes

Todos los componentes tendrán información estática, es decir, no informaremos datos a partir de ningún array

Las imágenes, a diferencia de los ejercicios anteriores, las colocaremos en una carpeta **img** pero dentro de la carpeta **public**, de esta forma, podremos acceder a ellas sin necesidad de utilizar el import o el require. Solo hay que indicar su ruta relativa tal como vemos en la estructura anterior



ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.1: Crear componente inicio.js

Dentro de la carpeta componentes crearemos también el componente de inicio en donde construiremos de forma dinámica todas las imagenes de los micros a partir de array (usando el método map) y añadiendo, a cada una de ellas la directiva <Link> para poder acceder a cada componente al pulsar sobre cada imagen:

- Incorporamos los import del array de micros y de la directiva Link

```
import micros from './Micros'
import { Link } from 'react-router-dom'
```

- Confeccionamos los enlaces de las imágenes de forma dinámica

```
<div className='micros'>
        {micros.map((micro, index) =>
          <div key={index} className='micro'><Link to={micro.path}><img
          src={`../img/${micro.imagen}`}/></Link></div> )}
</div>
```

ACTIVIDAD

Actividad 8.1: Crear componente inicio.js

Y añadimos un poco de css en **App.css**

```
.micros {display: flex;flex-wrap: wrap;justify-content: space-evenly;margin-top:50px}

.micro {width: 26%; cursor:pointer}
```

Oric Atmos

ACTIVIDAD

Actividad 8.1: Crear las rutas en App.js

Modificamos el componente principal **App.js** para crear todas las rutas de la aplicación

```
<div className="App">
   <BrowserRouter>
    <Routes>
      <Route path='/' element={<Inicio/>}/>
      <Route path='/inicio' element={<Navigate to="/"/>}/>
      <Route path='/dragon' element={<Dragon/>}/>
      … resto de rutas de los componentes ...
    </Routes>
   </BrowserRouter>
</div>
```

NOTA: Recordad realizar los import necesarios

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.1: Crear las rutas en App.js

Ahora deberíamos comprobar que al pulsar sobre cada imagen se carga el componente correspondiente

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.1: Confeccionar la barra de navegación Nav.js

Por último crearemos la barra de navegación para poder acceder a cada componente desde ella. Para ello crearemos el componente **componentes/Nav.js**:

```
<nav>
    <NavLink to={'/'}>Inicio</NavLink>
    <NavLink to={'/dragon'}>Dragon</NavLink>
    <NavLink to={'/newbrain'}>New Brain</NavLink>
    <NavLink to={'/oric'}>Oric</NavLink>
    <NavLink to={'/spectrum'}>Spectrum</NavLink>
    <NavLink to={'/toshiba'}>Toshiba</NavLink>
    <NavLink to={'/zx81'}>ZX 81</NavLink>
</nav>
```

Y un poco de css:

```
nav a {text-decoration: none; color: white; background-color: black; padding: 10px 20px;}
nav a:hover {background-color: brown;}
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.1: Confeccionar la barra de navegación Nav.js

También añadiremos la mejora de usabilidad para que el enlace pulsado aparezca resaltado con respecto al resto.

Unicamente hay que añadir la class .active en **app.css**:

```
.active {color:gold; background-color: blueviolet;}
```

Y para que se muestre la barra hay que añadir el componente <Nav> en **App.js**

```
<BrowserRouter>
    <Nav/>
    <Routes>
    .../...
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.2:

Vamos a refactorizar el código para hacerlo mucho más robusto.

El problema que tenemos ahora es que si tenemos que añadir o eliminar algún micro de nuestra lista de micros, tenemos que modificar todo el código para añadir o eliminar componentes y opciones de la barra de navegación

Imaginemos que los micros los obtenemos de una base de datos de forma que el número de ellos puede ser muy cambiante.

Necesitamos un sistema mucho más robusto que sea totalmente independiente del número de elementos que tenemos

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.2: Componente principal App.js

Modificaremos el componente principal para utilizar únicamente tres rutas: las dos rutas que apuntan al componente inicio y una ruta a un componente nuevo <Micros> que crearemos más adelante y que será el único componente que utilizaremos para mostrar cada uno de los micros:

```
<BrowserRouter>
    <Nav/>
    <Routes>
      <Route path='/' element={<Inicio/>}/>
      <Route path='/inicio' element={<Navigate to="/"/>}/>
      <Route path="/micros/:micro" element={<Micro/>}/>
    </Routes>
</BrowserRouter>
```

NOTA: Observad como enviaremos a la ruta /micros un parámetro :micro que será el nombre del micro a consultar

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.2: Componente barra de navegación Nav.js

Modificaremos el componente de la barra de navegación para confeccionar todos los enlaces de forma dinámica a partir de los elementos que tengamos en el array Micros

Importamos el array:

```
import micros from './Micros'
```

Confeccionamos de forma dinámica todos los enlaces:

```
<nav>
    <NavLink to={'/'}>Inicio</NavLink>
    {micros.map((micro, index) =>
      <NavLink key={index} to={`/micros/${micro.path}`}>{micro.modelo}</NavLink>
    )}
</nav>
```

NOTA: Observad como informamos el parámetro de la ruta /micros con el nombre del micro obtenido del atributo path del array

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.2: Componente Inicio.js

Modificaremos también el componente **inicio.js** que, aunque ya lo tenemos adaptado para que muestre todos los micros de forma dinámica, tenemos que cambiar la ruta ya que ahora hay que enviar el parámetro con el nombre del micro a consultar

```
<div className='micros'>
      {micros.map((micro, index) =>
          <div key={index} className='micro'><Link to={micro.path}>
          <div key={index} className='micro'><Link to={`/micros/${micro.path}`}>
          <img src={`../img/${micro.imagen}`} alt={micro.modelo}/></Link></div>
      )}
</div>
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.2: Nuevo componente Micro.js

Y, por último, creamos el nuevo componente **Micro.js** que substituirá a los seis componentes que teníamos antes (y que podemos eliminar):

En este componente tendremos que recuperar el parámetro que enviamos por la url con el nombre del micro por lo que necesitaremos el hook useParams

```
import { useParams } from 'react-router-dom'
```

Recogemos el dato que nos llega en el parámtro utilizando:

```
const {micro} = useParams() //desestructuramos el dato
```

NOTA: Recordad que con el uso de llaves {micro} desestructuramos el dato que nos llega como un array de objetos al utilizar useParams.

Con const micro tendríamos un objeto: `{micro: 'newbrain'}`

Con const {micro} tenemos directamente el valor: `newbrain`

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.2: Nuevo componente Micro.js

Necesitamos acceder al array micros para recuperar los datos que necesitamos mostrar en la ficha: modelo del micro y imagen a partir del dato que nos llega como parámetro al componente y que corresponde al atributo path del array

```
path: 'spectrum',
modelo: 'ZX Spectrum',
imagen: 'sinclair_zx_spectrum.png'
```

Para buscar el elemento en el array que corresponda al valor del path que nos llega en la url utilizaremos el método find de JS:

```
const datos = micros.find((item) => item.path === micro)
```

ALCYON
IT Servicios

ACTIVIDAD

Actividad 8.2: Nuevo componente Micro.js

Y ya tenemos un objeto datos con toda la información que necesitamos mostrar en pantalla:

```
<div className='ficha'>
    <img src={`../img/${datos.imagen}`} alt={datos.modelo}/>
    <h2>{datos.modelo}</h2>
</div>
```

Inicio | ZX Spectrum | ZX 81 | Dragon 32 | New Brain | Oric Atmos | Toshiba MSX

Dragon 32

Aviso Legal

David Alcolea
david-alcolea@alcyon-it.com
www.alcyon-it.com