Front End III

Crear rutas parametrizadas

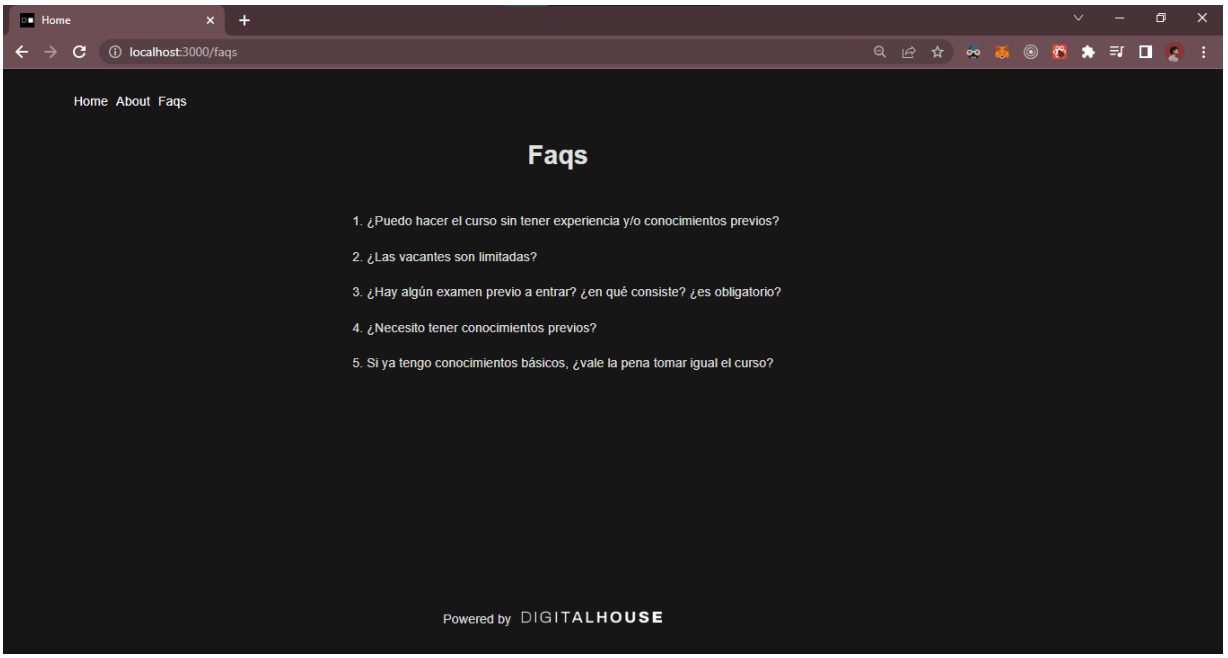
Pasemos a desarrollar nuestro ejemplo. Esta es la información con la que estamos trabajando:

```
const preguntas = [

    {

            id:1,

            title: "¿Puedo hacer el curso sin tener experiencia y/o conocimientos previos?",

         descripción: "Sí. En Digital House puedes aprender desde cero. Según el curso al que te anotaste, vamos a enviarte contenido previo online..."

    },

    {

       id:2,

        title: "¿Las vacantes son limitadas?",

        descripción: "Sí. Tienen una capacidad máxima de entre 50 y 75 personas, dependiendo el curso."

    },

   //...

]
```

Vamos ahora a renderizar esta lista con el método **map**.

```
const Faqs = () => {
  return (
    <div>
        <h1>Faqs</h1>
        <ol>
          {preguntas.map(pregunta => (
            <li key={pregunta.id}>
              {pregunta.title}
            </li>
          ))}
        </ol>
    </div> )
}
export default Faqs
```

¿Cómo se ve en nuestra página?



Ahora, ¿cómo hacemos para que cada pregunta sea una ruta? Como ya habrás adivinado, debemos envolver cada elemento de la lista con el componente **Link**:

```
//...
 {preguntas.map(pregunta => (
   <Link key={pregunta.id} to={`${pregunta.id}`} >
     <li>
       {pregunta.title}
     </li>
   </Link>
 ))}
//...
```

Nótese que en el atributo **to** colocamos un valor propio o “personalizado” para cada pregunta con su ID.

Ahora bien, de la misma manera que anteriormente definimos los Links (que por sí solos no hacen nada), ahora debemos ir a nuestro archivo **App** y definir el componente que queremos mostrar para estas rutas especiales:

```
//index.js o main.js

ReactDOM.createRoot(document.getElementById('root')).render(

  <React.StrictMode>

    <BrowserRouter>

      <Routes>

          <Route path='/' element={<App/>}  >

            //...

            <Route path='faqs/:id' element={<Faq/>}  />

          //...

          </Route>

      </Routes>

    </BrowserRouter>

  </React.StrictMode>

)
```

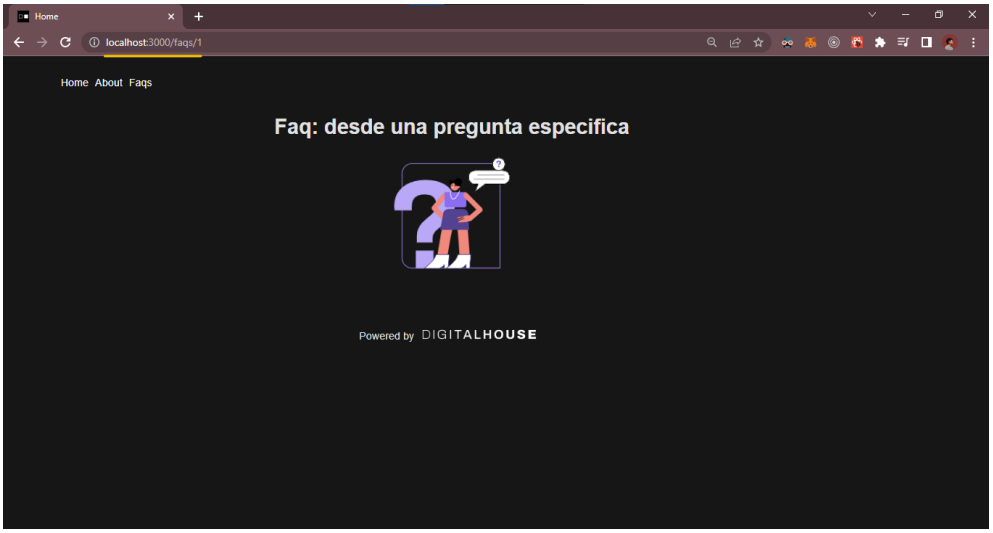
Nuestra nueva ruta tiene un aspecto diferente al resto. Veámosla en detalle:

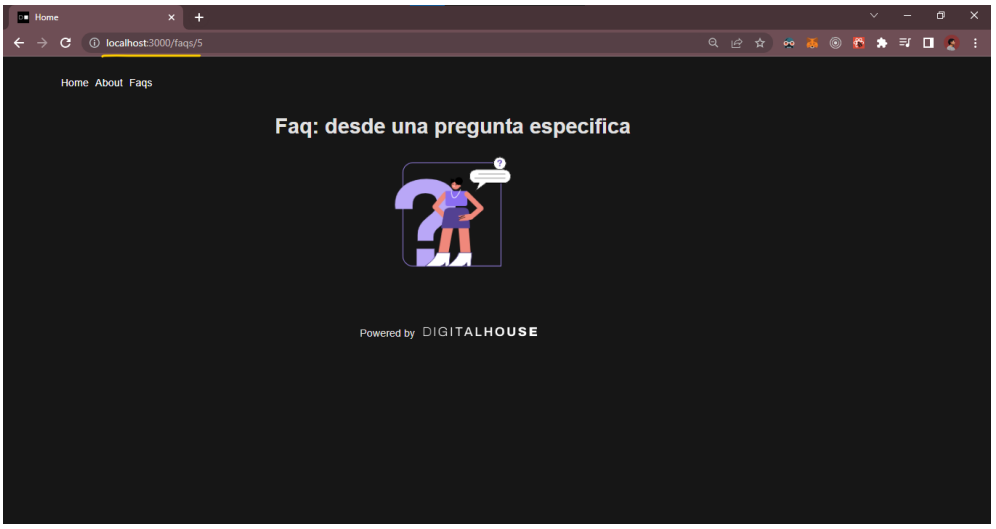
```
<Route path='faqs/:id' element={<Faq/>}  />
```

Como habrás notado, agregamos el signo "**:**", seguido del valor que tomará la ruta. De esta manera, lo que estamos haciendo es decirle a React Router que ese **valor que va a continuación de /faqs/ va a ser dinámico**, es decir, que puede variar.

Ahora, **nuestra nueva ruta va a matchear con cualquier valor que siga a /faqs/**, por ejemplo, para cada pregunta específica que tengamos: **/faqs/1**, **/faqs/2** o **/faqs/3**. Solo debemos ser cuidadosos a la hora de nombrar, dado que esta ruta también va a matchear cualquier otro contenido, como **/faqs/digital**, **/faqs/rojo** o hasta **/faqs/home**.

Veamos cómo se ve nuestra página cuando cambiamos a una ruta específica de cada pregunta:





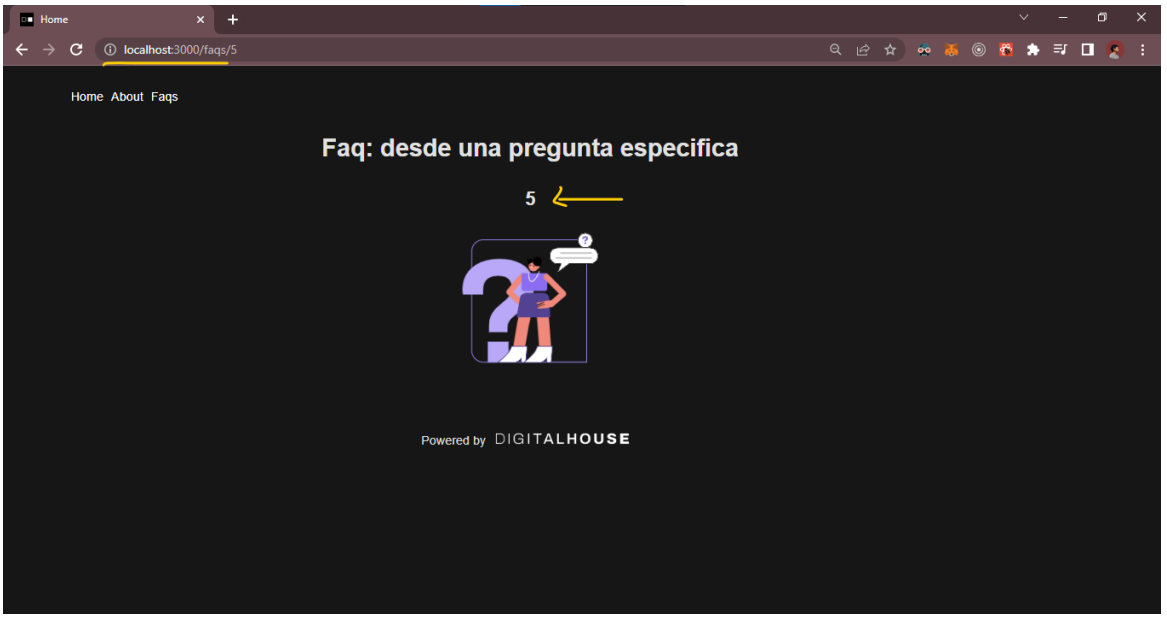
¿Pero de qué nos sirve esto si seguimos mostrando un mismo contenido independientemente de la pregunta a la que redirijamos a las personas usuarias? De la misma manera que le dijimos a React Router que ese parámetro de la ruta será dinámico, también podemos decirle que nos **devuelva qué valor está tomando ahora ese parámetro en la URL**. ¿Cómo lo hacemos? Utilizaremos un Hook propio de la librería, llamado **useParams()**:

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


const Faq = () => {
  const params = useParams()
  return (
    <div>
        <h1>Faq: desde una pregunta específica</h1>
        <h2>{params.id}</h2>
        <img src={faq} alt="faq"/>
    </div>
  )}
```

Este Hook nos **devolverá un objeto con el cual tendremos acceso al valor dinámico de nuestra ruta**. Para acceder al mismo, al igual que en cualquier objeto, accedemos a la información mediante su key. En este caso, la información estará dentro de la propiedad **id**, dado que así lo nombramos cuando definimos la ruta.

```
/faqs/:id -> params.id
```

Veámoslo funcionando en nuestra web:

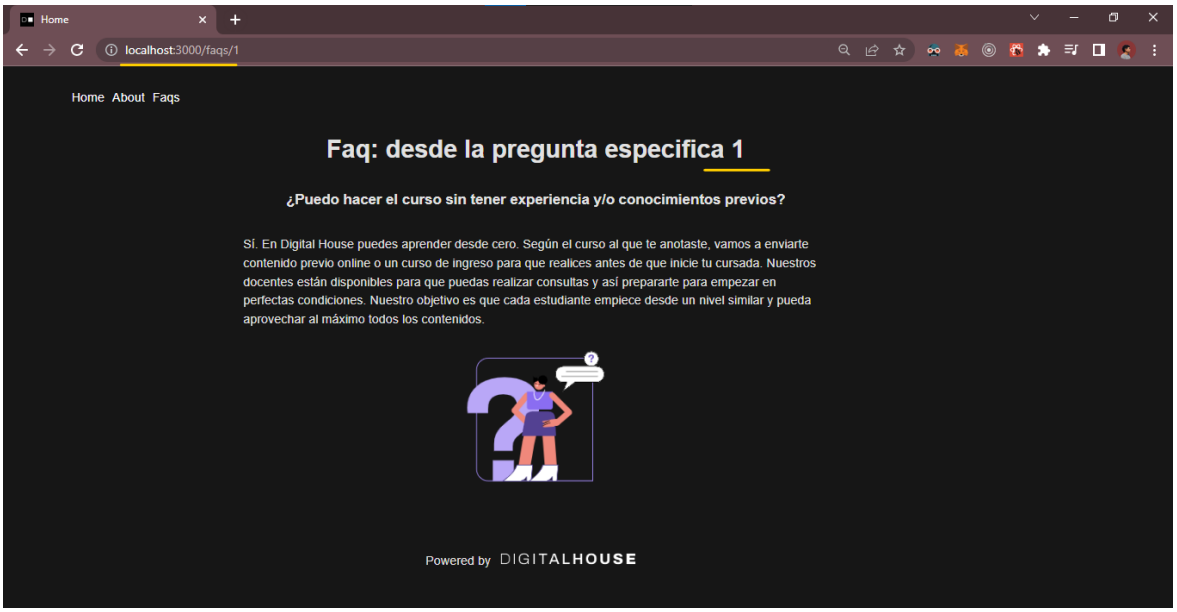


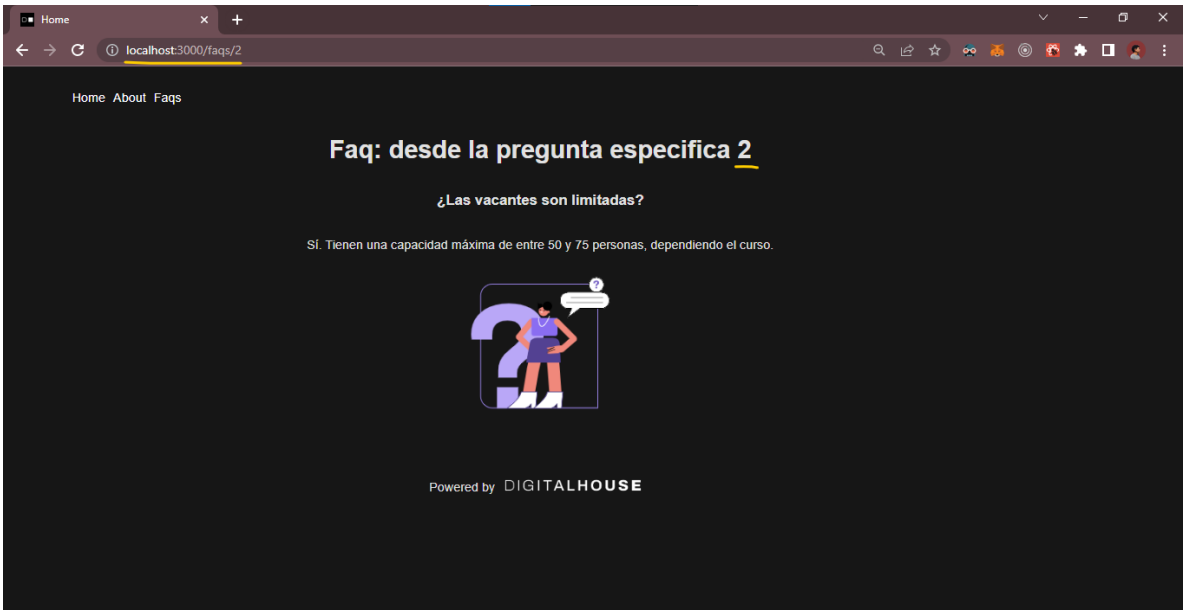
Ahora que tenemos este valor, podemos empezar a jugar y traer —por ejemplo— la pregunta específica que matchee con ese ID y renderizarla:

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


const Faq = () => {
  const params = useParams()
  //Utilizamos parseInt solamente para pasar el id al tipo number
  const pregunta = preguntas.find(pregunta => pregunta.id === parseInt
(params.id))

  return (
    <div>
```

```
      <h1>FAQ: desde la pregunta específica {params.id}</h1>
      <section>
        <h3>{pregunta?.title}</h3>
        <p>{pregunta?.descripcion}</p>
      </section>
      <img src={faq} alt="faq"/>
    </div>
  )}
```

Veámoslo en nuestra web:





Bonus track

También podemos definir una ruta predeterminada que se muestre en el caso de que el usuario quiera dirigirse a una URL inexistente. Hacerlo es muy sencillo, solo debemos agregar el símbolo "*" al atributo **path** de nuestra ruta y ya estará funcionando esta implementación:

```
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter>
      <Routes>
          <Route path='/' element={<App/>}  >
            <Route index path='home' element={<Home/>}  />
            <Route path='about' element={<About/>}  />
            <Route path='faqs' element={<Faqs/>}/>
```

```
          <Route path='faqs/:id' element={<Faq/>}  />
        </Route>
        <Route path='*' element={<ErrorComponent/>} />
    </Routes>
  </BrowserRouter>
</React.StrictMode>)
```