React Router

La première pour mettre en place plusieurs écran c'est de choisir une librairie de gestion des écrans. Ce qu'on vas découvrir, c'est la librairie officiel :

React Router

Installation

Pour pouvoir utiliser le react router :

Version web :

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

Version native :

```
npm i react-router-native
```

Mettre en place le Router

Pour commencer à utiliser le react-router il faut tou d'abord utiliser la composant : `<BrowserRouter />`

Exemple en version web :

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

export default function App() {
  return (
    <>
      <BrowserRouter></BrowserRouter>
    </>
  )
}
```

Exemple en version native :

```
import { NativeRouter } from 'react-router-native'

export default function App() {
  return (
    <>
      <NativeRouter></NativeRouter>
    </>
  )
}
```

Utiliser des Route et des Link

Pour pouvoir créer plusieurs écrans dans notre application (un écran de Connection, et un écran d'inscription). Il faut utiliser les composants `<Routes />`, `<Route />` et `<Link />` :

Exemple :

```
import Home from './Home'
import Register from './Register'
import Connect from './Connect'
import { BrowserRouter, Routes, Route, Link } from 'react-router-dom'

export default function App() {
  return (
    <>
      <BrowserRouter>
        {/* Permet de mettre en place un série de routes */}
        <Routes>
          {/* Mettre en place les routes de l'application en utilisant l'URI */}
          <Route path="/" element={<Home />}>
          <Route path="/inscription" element={<Register />}>
          <Route path="/connexion" element={<Connect />}>
        </Routes>

        {/* Création d'un menu */}
        <nav>
          {/* Lien de navigation */}
          <Link to="/">Accueil</Link>
          <Link to="/inscription">Inscription</Link>
          <StyledLink to="/connexion">Connexion</StyledLink>
        </nav>
      </BrowserRouter>
    </>
  )
}

/**
 * Pour stylisé une balise Link
 */
export const StyledLink = styled(Link)`
  background-color: red;
`
```

Utiliser des routes dynamiques

Dans la plupart des applications d'aujourd'hui, il éxiste un système d'URI dynamique. Exemple, l'écran `/liste/petite-course` c'est le même que l'écran : `` `/liste/à-faire-demain ``, c'est à dire que ces deux routes affiche le même composant. C'est seulement l'URI qui contient des données différente.

Pour mettre en place une route dynamique il faut créer un paramètre de route :

```
/*
  Grace au `:listName`, je déclare un paramètre de route dynamique
  nommé `listName`
*/
<Route path="/liste/:listName" element={<TodoList />} />
```

Récupérer le paramètre de route dans un composant

Pour récupérer les paramètres dans un composant on utilise le hook : `useParams`

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

export default function TodoList() {
  // J'ai besoin de récupérer le paramètre dynamique de route : `listName`
  const { listName } = useParams()

  // Code nescessaire pour récupérer la liste ...
}
```

Rediriger dans un composant en utilisant navigate

Pour pouvoir rediriger vers d'autres composant en fonction de certaines condition,
il faut utiliser la fonction `navigate` que l'on obtient avec le hook : `useNavigate`.

> La fonction `navigate` est IMPURE !!!! Attention, on ne peut pas l'utiliser diréctement dans la fonction du composant, car cette fonction composant est PURE ! On préfére utiliser un effet

```
import { useNavigate, useParams } from 'react-router-dom'
import { useState } from 'react'

export default function TodoList() {
  // Crééation d'un état qui permet de savoir si le composant est en train
  // de se charger (loading) ou si il est prêt à s'afficher
  const [loading, setLoading] = useState(true)
  // Création d'un autre état contenant la liste de choses à faire
  const [list, setList] = useState(null)
  // On récupére le nom de la liste contenue dans les paramètres de routes
  const { listName } = useParams()
  // On importe la fonction de navigation
  const navigate = useNavigate()

  useEffect(() => {
    // Récupération de la liste avec le nom passé en paramètre de la route
    let retrievedList = getList(listName)

    // Si je n'ai pas réussi à récupérer la liste
    if (!retrievedList) {
      // On navigue vers une autre page
      navigate('/')

      return
    }
```

```
    // Ici, j'ai bien une liste, je la met donc dans mon état
    // afin de rafraichir l'affichage
    setList(getList(listName))
    // On désactive le chargement
    setLoading(false)
  }, [])

  // Si je suis en train de charger
  if (loading) {
    return <p>Chargement en cours ...</p>
  }

  return (
    <>
      <h1>Ma Liste</h1>
      {list.map(todo => (
        <p>{todo}</p>
      ))}
    </>
  )
}
```