8. REACT-ROUTER

Mappa un URL verso un componente specifico

Browser

- window.location (Location API)
- window.history (History API)
  - pushState(href)—aggiunge una nuova location allo stack
  - replaceState(href)—sostituisce la location attuale
  - back()—naviga alla precedente location nello stack
  - forward()—naviga alla successiva location nello stack
  - go(index)—naviga ad una specifica location nello stack

history

- Unisce le funzionalità di window.location e window.history in un'unica istruzione
- history.goBack()
- history.goForward()
- history.go(-3)
- history.replace({ pathname: '/go-here-instead' })
- history.push({ pathname: '/new-place' })

const browserHistory = createBrowserHistory()
const hashHistory = createHashHistory()

In React viene usata una **hash history**

L'unico file statico è **index.html**

- Viene creata una navigazione basata sugli hash
- http://www.example.com/#/login
- http://www.example.com/#/main

```
import {createHashHistory} from 'history';

export const history = createHashHistory();
```

Esercizio

- creare un file **utils/history.js**
- Configurare la history del progetto
- Importare **utils/history.js** in App.js

I componenti sono figli del Router secondo una logica **HoC** (High Order Components)

Struttura

Router

```
import { Router } from 'react-router-dom'
import history from '../services/history'

ReactDOM.render((
  <Router history={history}>
    <App />
  </Router>
), document.getElementById('root'))
```

Non necessario con Redux

Route

```
<Route path='/roster' component={Roster}/>
```

- **path**
- **strict (bool)**
  - /path/ - /path/ -> true
  - /path/ - /path -> false
  - /path - /path/whatever -> false
- **exact (bool)**
  - /path - /Path -> false
- **sensitive(bool)**
- params
- props.match.params

Esercizio

- **npm i --save react-router-dom**
- Importare Router e Route nel progetto
- Effettuare un Refactor di App.js includendo i componenti per il Routing, spostando il rendering della lista in un componente separato
- Definire le Route applicative per la scherma di elenco / aggiunta / modifica

Link

NavLink

- activeClassName
- Navigazione verso altre Routes
- history.push({ pathname: '/new-place' })

```
<Link to="/about">About</Link>
```

Switch

```
th='/' component={Home}/>
oster' component={Roster}/>
chedule' component={Schedule}/>
t={NotFoundPage} />
```

```
<Route component={NotFoundPage} />
```

Utilizzato per gestire le varie condizioni del Routing, e definire una Route di default

Esercizio

- Definire un componente NavBar e inserire i Link di navigazione verso la schermata di elenco e la schermata di aggiunta
- Includere le Route in un componente Switch
- Definire una Route di default che renderizzi un componente Not Found

Redirect

Da utilizzare per reindirizzare l'utente prima che venga risolta la Route richiesta

```
const fakeAuth = {
  isAuthenticated: false
};

const PrivateRoute = ({ component: Component, ...rest }) => (
  <Route
    {...rest}
    render={props =>
      fakeAuth.isAuthenticated ? (
        <Component {...props} />
      ) : (
        <Redirect
          to={{
            pathname: "/login",
            state: { from: props.location }
          }}
        />
      )
    }
  />
);
```

Esercizio

- Creare un componente Login che contenga i campi Username e Password, ed un bottone per il login
- Modificare le Route in modo che se l'utente non è autenticato, venga reindirizzato alla route di Login
- Creare una procedura che esegua un fake login e reindirizzi l'utente alla route principale