Seminar 11

Crearea de rute în React

În React, crearea de rute ne ajută în legătura cu modul în care o aplicație web navighează între diferitele componente sau pagini. Atunci când construim o aplicație mai complexă, aceasta poate avea mai multe pagini sau secțiuni care trebuie afișate în funcție de interacțiunea utilizatorului.

❖ React Router

React în sine **nu oferă funcționalități native pentru gestionarea rutării**, dar există biblioteci precum **React Router** care ne permit să implementăm ușor sistemul de rutare într-o aplicație React. Cu ajutorul React Router sau altor soluții similare, putem defini rutele aplicației, adăuga link-uri între aceste rute și gestiona comportamentul afișării componentelor în funcție de ruta curentă a aplicației.

Principalele componente:

1. **BrowserRouter** – este o componentă care se ocupă de gestionarea rulării aplicației web într-un mediu de browser. Acesta utilizează API-ul de istoric al browserului (**window.history**) pentru a asigura o experiență de navigare fluidă și sincronizată între URL-ul afișat în bara de adrese a browserului și starea aplicației tale React.
2. **Link –** o componentă care permite utilizatorului să navigheze la o altă pagină apăsând click fără să se dea refresh la pagină. Acesta actualizează doar URL-ul, astfel încât React să poată decide ce componentă trebuie afișată în funcție de noul URL. Această componentă se folosește în locul clasicei ancore pentru a nu se mai da refresh la pagină.
3. **Route** - o componentă care definește o rută în aplicație. Atunci când URL-ul se potrivește cu calea specificată într-o componentă Route, acesta va afișa componenta asociată.
4. **Routes –** această componentă părinte reprezintă locul în care sunt create toate rutele într-un mod ierarhic, făcând mai ușor de înțeles structura aplicației.
5. **Hook-uri** – ne oferă facilități pentru gestionarea rutelor, permițând dezvoltatorilor să acceseze și să manipuleze diverse informații

❖ Instalare React Router

Deschide terminalul și rulează următoarea comandă:

```
npm install react-router-dom
```

❖ React Router Hooks

Lista principalelor hook-uri:

1. **useParams** - este un hook care permite componentelor funcționale dintr-o aplicație să acceseze parametrii specifici dintr-o rută.

Atunci când avem o rută definită cu parametri în React Router, cum ar fi **'/users/:id'**, putem utiliza **useParams** pentru a accesa valoarea parametrului id din URL. Acest hook este util atunci când vrem să extragem și să utilizăm valorile din calea URL-ului în cadrul componentei.

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

const UserProfile = () => {
  // Extrage parametrul 'id' din URL
  const { id } = useParams();

  return (
    <div>
      <h2>User Profile</h2>
      <p>User ID: {id}</p>
    </div>
  );
};
```

2. **useNavigate** - este un hook care furnizează o funcție de navigare pe care o putem apela pentru a realiza navigarea programatică în cadrul aplicației.

```
import { useNavigate } from 'react-router-dom';

const MyComponent = () => {
  const navigate = useNavigate();

  const handleButtonClick = () => {
    // Navighează către '/Login'
    navigate('/Login');
```

```
  };

  return (
    <div>
      <button onClick={handleButtonClick}>Login</button>
    </div>
  );
};
```

3. **useLocation** - este un hook care accesează obiectul **location**. Obiectul **location** conține informații despre locația curentă a aplicației, inclusiv calea URL, căutarea, starea și alte detalii relevante.

```
import { useLocation } from 'react-router-dom';

const MyComponent = () => {
  const location = useLocation();

  console.log('Pathname:', location.pathname);
  console.log('Search:', location.search);
  console.log('Hash:', location.hash);
  console.log('State:', location.state);
};

export default MyComponent;
```

4. **useSearchParams** - este un hook introdus în React Router v6 care furnizează un mod ușor de a accesa și manipula parametrii de căutare (query parameters) din URL. Acesta este util atunci când lucrăm cu query parameters în cadrul unei componente React și dorim să accesăm și să actualizăm acești parametri.

```
import { useSearchParams } from 'react-router-dom';

const MyComponent = () => {
  const [searchParams, setSearchParams] = useSearchParams();

  // Citirea parametrului 'name' din URL
  const nameParam = searchParams.get('name');

  // Actualizarea valorii parametrului 'page' în URL
  setSearchParams({ page: 2 });
};

export default MyComponent;
```

❖ Folosirea cheilor de environment în React

În React, putem utiliza variabilele de mediu pentru a gestiona setările de configurare sau informațiile sensibile fără a le introduce direct în cod.

Primul pas este crearea unui fișier de mediu în rădăcina proiectului React. Numele fișierului ar trebui să înceapă cu .env urmat de numele mediului (de exemplu, .env.development sau .env.production).

Exemplu:

```
REACT_APP_API_URL=https://api.exemplu.com
```

Note:

1. Trebuie să ne asigurăm că prefixăm variabilele cu **REACT_APP_** pentru a ne asigura că sunt recunoscute și pentru a evita conflicte cu alte variabile.
2. În timpul build-ului, **variabilele de mediu sunt citite și înlocuite în codul sursă React**. Asta înseamnă că, la runtime, **aceste valori vor fi accesibile în cadrul aplicației**. Dacă ai chei importante în aceste variabile de mediu, ele vor fi expuse în sursa codului transpus, iar oricine ar inspecta sursa aplicației va putea vedea aceste chei.
3. Orice informație stocată în aplicația React este accesibilă la nivel de client (browser). Chiar dacă cheile sunt utilizate în mod corect în cadrul aplicației, ele pot deveni accesibile și exploatabile de către utilizatorii care inspectează sursa în browser.