JUNIORDev DIGITALNA DALMACIJA

CRUD operacije > Brisanje podataka

Brisanje podataka

Osim zahtjeva za dohvat i spremanje podataka, poslužitelju također možemo poslati i zahtjev za brisanje nekog podatka. U tom slučaju se koristi ***DELETE*** tip zahtjeva.

Kako ne bi gubili previše vremena za prilagodbu sučelja, napraviti ćemo novu komponentu `<Brisanje />` sa jednim poljem za unos koje će nam služiti za brisanje podataka. Jednom kada se upoznamo sa načinom na koji funkcionira ova vrsta zahtjeva, možemo doraditi samo sučelje aplikacije.

Brisanje.jsx

```
import axios from "axios";
import { useState } from "react";

function Brisanje(props) {
  const [idPodatka, postaviId] = useState(0);

  function brisiPodatak() {
    console.log(`Brišem podatak broj ${idPodatka}`);
  }
  return (
    <div>
      <h2>Brisanje podataka</h2>
      <div>
        <label>
          Unesite ID podatka:
          <input
            type='number'
            name='id'
            value={idPodatka}
            onChange={e => postaviId(e.target.value)}
          />
        </label>
      </div>
      <button onClick={brisiPodatak}>Obriši rezervaciju</button>
```

```
      </div>
    );
  }

  export default Brisanje;
```

Svi početni elementi komponente bi nam trebali biti poznati - jedna kontrolirana komponenta sa pripadajućom varijablom stanja i pomoćna funkcija za `onClick` događaj. Polje za unos nam služi kako bi mogli unijeti ID podatka kojeg želimo izbrisati sa poslužitelja.

Slanje DELETE zahtjeva

Slanje DELETE zahtjeva je prilično jednostavno, pogotovo kada koristimo *Axios*. Sve što je potrebno je pozvati ugrađenu `axios.delete()` metodu i upisati ispravnu adresu podatka (resursa) kojeg želimo izbrisati.

Za točnu adresu bi naravno trebalo pogledati dokumentaciju poslužitelja. U našem slučaju lokalni poslužitelj koristi REST arhitekturu što znači da svaki resurs ima jedinstvenu putanju. U našem slučaju to znači da zahtjev za brisanje moramo poslati na adresu `/rezervacije/:id`, gdje umjesto `:id` šaljemo brojčanu vrijednost koja odgovara ID podatku kojeg želimo izbrisati. Implementacija funkcije izgleda ovako:

```
function brisiPodatak() {
  console.log(`Brišem podatak broj ${idPodatka}`);
  axios
    .delete(`http://localhost:3001/rezervacije/${idPodatka}`)
    .then(rez => console.log(rez));
}
```

Sadržaj odgovora ponovno ovisi o načinu na koji je poslužitelj implementiran. U ovom slučaju dobivamo odgovor 200 (OK), te odgovor nema nikakve dodatne podatke. Idući korak nam je ponovno osvježavanje prikaza.

Osvježavanje prikaza

Postupak osvježavanja je identičan kao i kod stvaranja nove rezervacije. Prvi korak nam je odabir načina na koji ćemo osvježiti stanje aplikacije - možemo ukloniti izbrisani podatak iz stanja komponente ili možemo poslati novi GET zahtjev poslužitelju kako bi dohvatili aktualne podatke. U svakom slučaju moramo iz glavnog dijela aplikacije komponenti poslati referencu na funkciju za promjenu stanja.

```
<Brisanje promjena={postaviRezervacije} />
```

Slanje novog zahtjeva je možda jednostavniji pristup pa ćemo krenuti od njega. Nakon potvrde uspješnog brisanja jednostavno ćemo poslati novi zahtjev i dobivene podatke spremiti kao novo stanje.

```
function brisiPodatak() {
  axios.delete(`http://localhost:3001/rezervacije/${idPodatka}`).then(rez => {
    axios
      .get("http://localhost:3001/rezervacije")
      .then(rez => props.promjena(rez.data));
  });
}
```

Implementacija je prilično jednostavna, samo moramo paziti na ispravnu sintaksu prilikom pisanja ugniježđenih zahtjeva. Ukoliko ste upoznati sa ***async/await*** sintaksom, to je elegantniji i pregledniji način pisanja.

```
async function brisiPodatak() {
  await axios.delete(`http://localhost:3001/rezervacije/${idPodatka}`);
  const rez = await axios.get("http://localhost:3001/rezervacije");
  props.promjena(rez.data);
}
```

Za osvježavanje popisa bez slanja zahtjeva možemo koristiti `filter()` metodu kako bi iz trenutnog stanja "izbacili" element sa odabranim ID-om.

```
  async function brisiPodatak() {
```

```
    await axios.delete(`http://localhost:3001/rezervacije/${idPodatka}`);
    props.promjena(stanje => stanje.filter(el => el.id != idPodatka));
  }
```

U ovom slučaju nam `filter()` metoda odgovara za naše potrebe ali pripazite kada je koristite u kombinaciji sa varijablama stanja jer ona radi samo **plitku** (***shallow***) kopiju objekta nad kojim se poziva, što može dovesti do neželjenih posljedica.

Zadatak

Ovaj način brisanja kroz polje za unos nije baš intuitivan, a postoji i problem provjere unosa. Bolja opcija bi bila kada bi u tablici uz svaku rezervaciju imali opciju brisanja te rezervacije.

Implementacija te funkcionalnosti ne zahtijeva nikakve nove funkcionalnosti pa je možete pokušati samostalno odraditi. Potrebno nam je:

- dodati novi stupac u zaglavlje tablice za prikaz
- dodati novi element u redak tablice - `<button>` element za brisanje
- pritiskom na tipku za brisanje potrebno je poslati DELETE zahtjev na poslužitelj
- pripazite da prilikom slanja šaljete ispravan ID u adresi
- nakon brisanja je potrebno osvježiti prikaz tablice

Last updated on March 21, 2023

‹ Slanje podataka

Promjena podataka ›